QoS-oriented service selection in Things-as-a-Service architectures

Enzo Mingozzi
University of Pisa, IT

ENSIMAG – Grenoble, November 17, 2015

Outline

- IoT horizontal platforms: centralized vs. distributed approach
  - the EU FP7-BETaaS solution: Things-as-a-Service on top of a "local cloud"
- Quality of Service support for M2M applications
  - the BETaaS QoS framework
- Optimized Thing Service selection
  - Exploit multiple TS equivalence to maximize system lifetime: the Agent Bottleneck Generalized Assignment Problem
- Ongoing & future work

Outline

- IoT horizontal platforms: centralized vs. distributed approach
  - the EU FP7-BETaaS solution: Things-as-a-Service on top of a "local cloud"
- Quality of Service support for M2M applications
  - the BETaaS QoS framework
- Optimized Thing Service selection
  - Exploit multiple TS equivalence to maximize system lifetime: the Agent Bottleneck Generalized Assignment Problem
- Ongoing & future work

Aknowledgement

- **EU-FP7 BETaaS (2012-15)**
  Building the Environment for the Things as a Service

http://www.betaas.eu

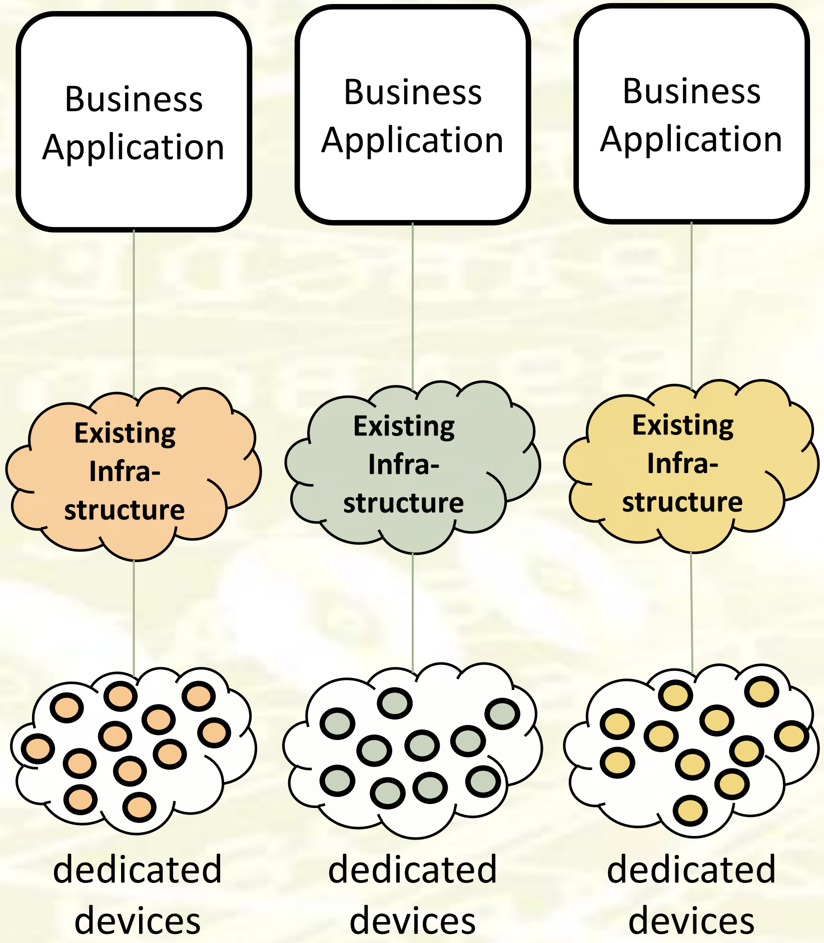
The BETaaS platform

- A reference framework enabling interoperable (**<u>horizontal</u>**) M2M application deployment
  - A **<u>distributed</u> runtime environment** based on a “local cloud” of gateways for the services to be deployed so as to fulfill high-level M2M applications
  - Content-centric Things-as-a-Service layered model
  - Built-in support for non-functional requirements (***<u>Quality of Service</u>***, big data management, dependability, security)
  - Seamless integration of existing IoT/M2M systems

- Implementation based on distributed OSGi
  - https://github.com/BETaaS/: setup, development tools, tutorials

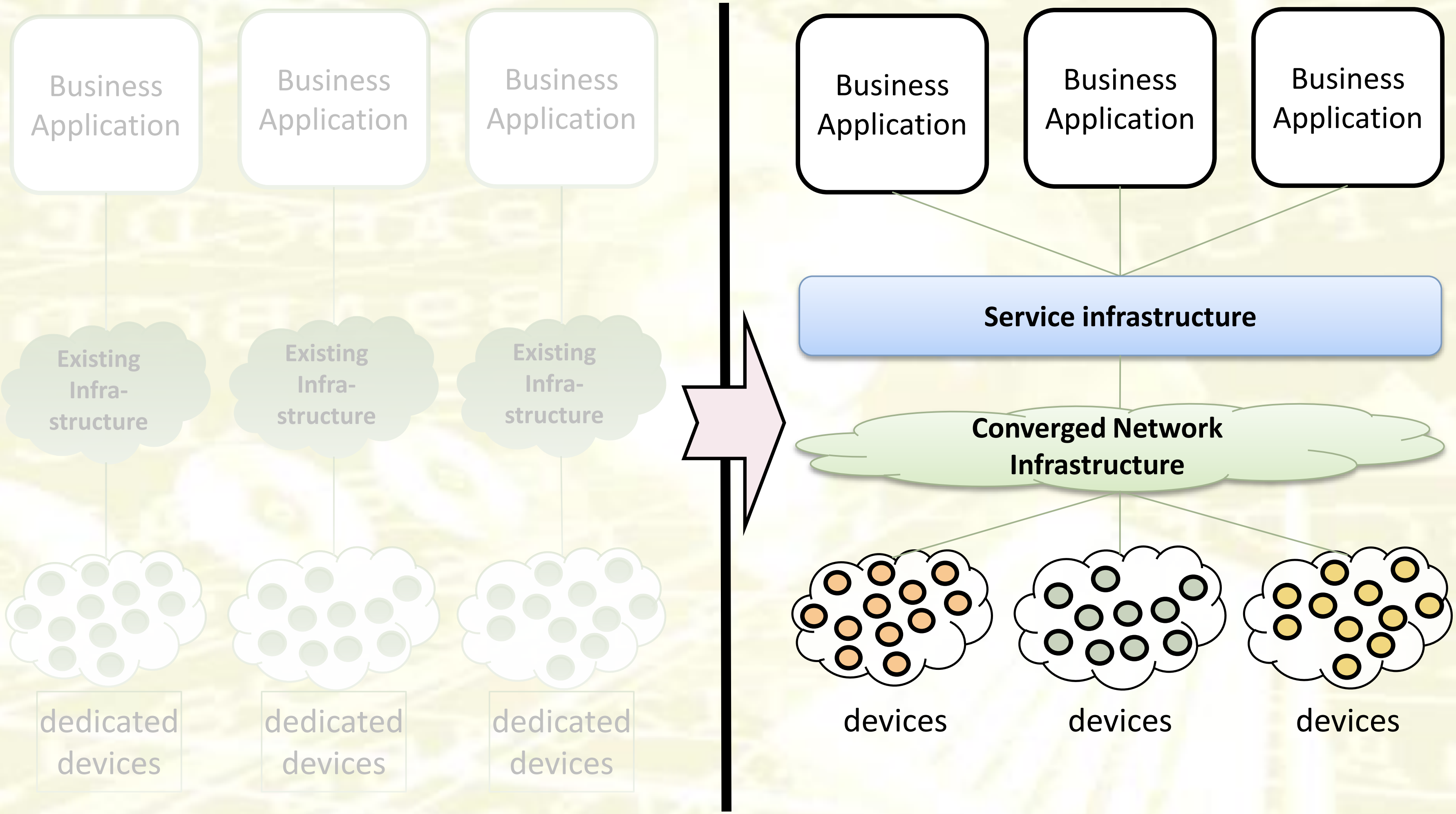
IoT: still a vision, or real already?

Vertical platforms

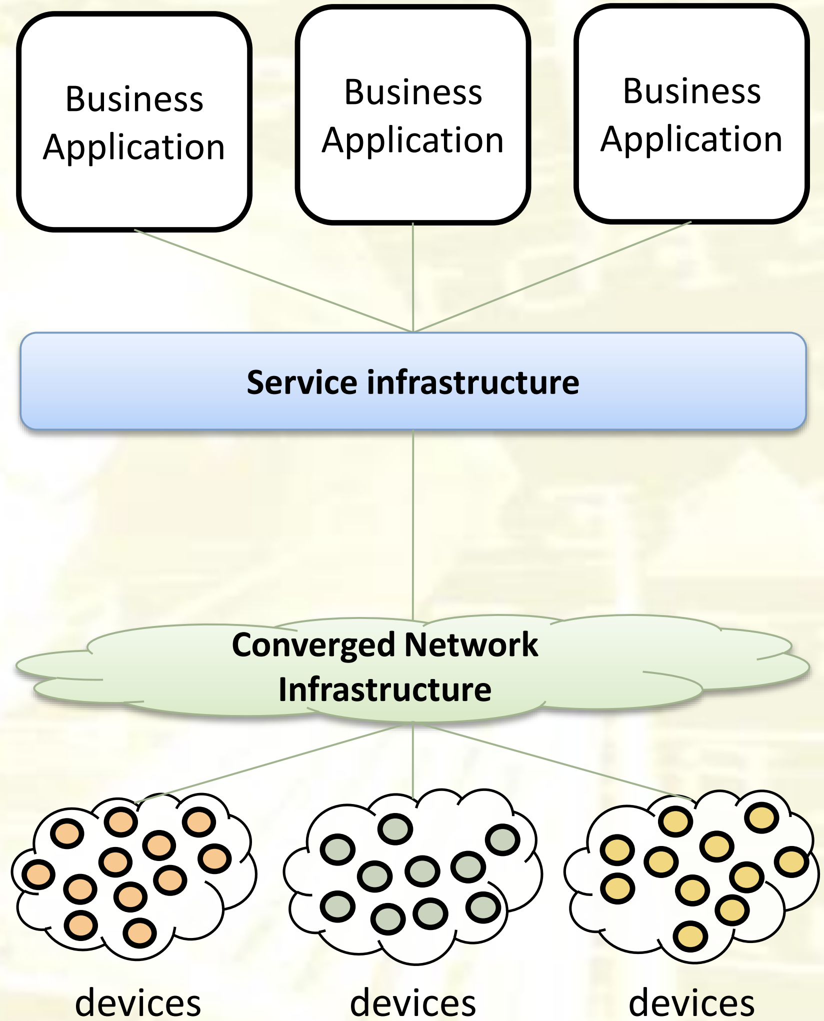


- Designed to serve one single purpose
- Inefficient: each device is dedicated to a single application
- Operate in isolation: no (or very limited) cooperation

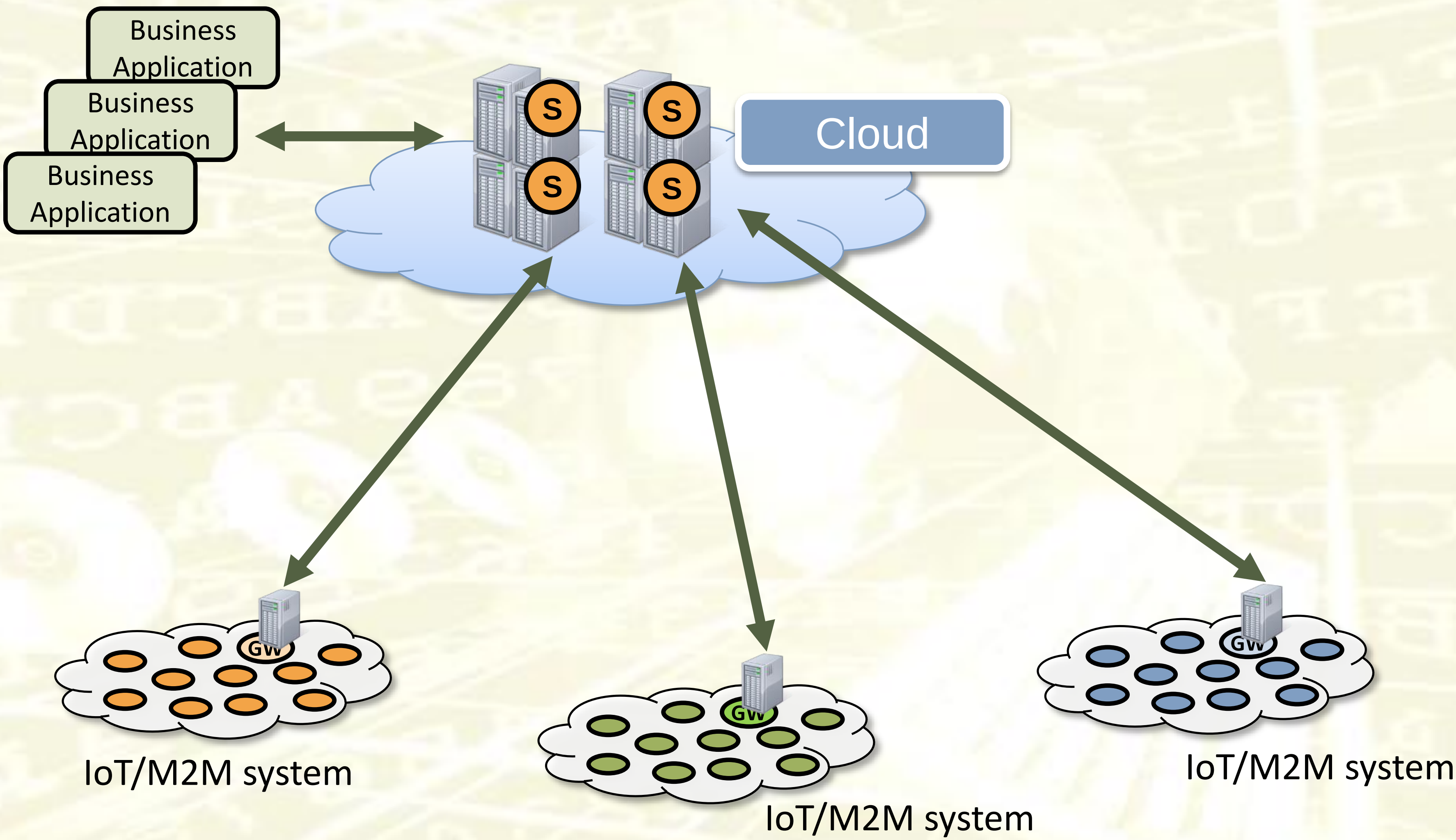
Horizontal approach



Horizontal approach

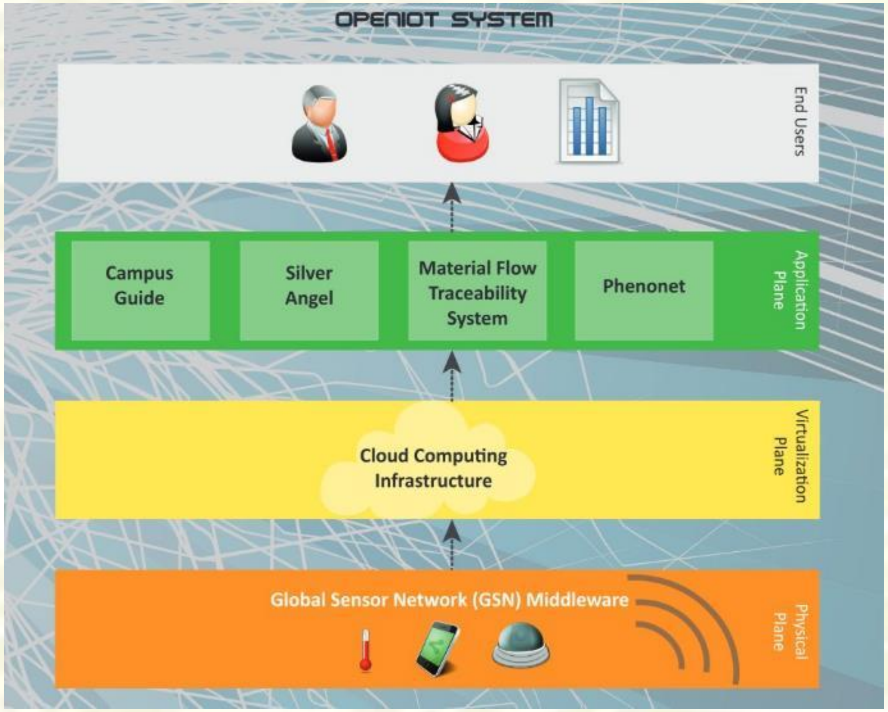


Cloud-based centralized platforms

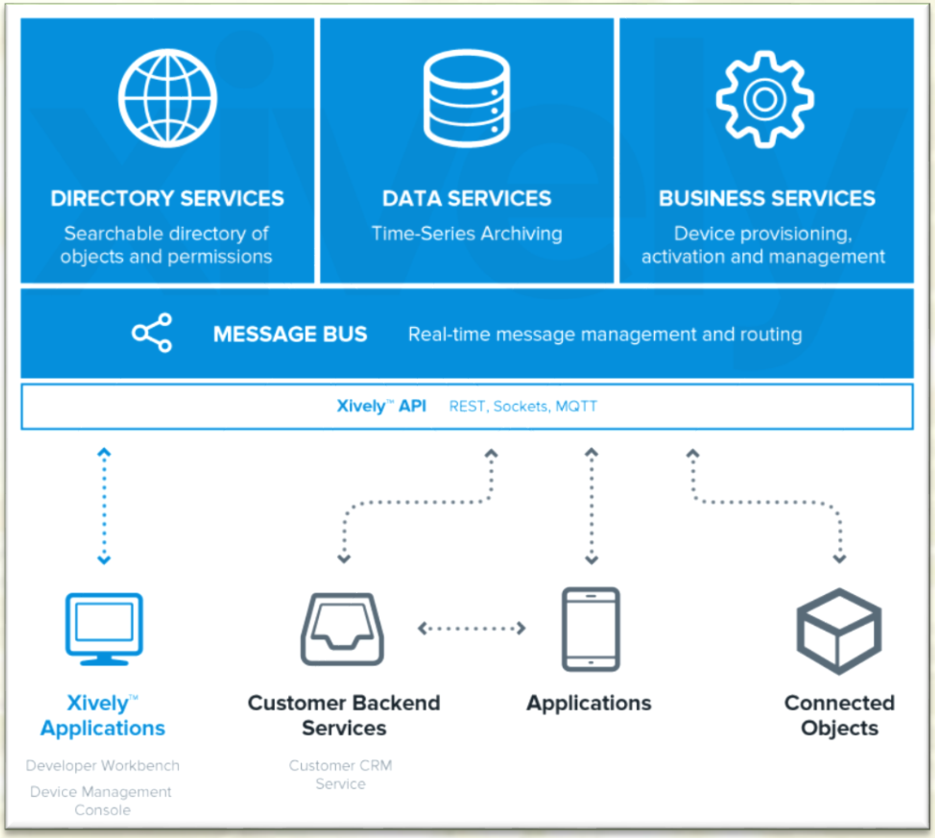


Cloud-based centralized platforms

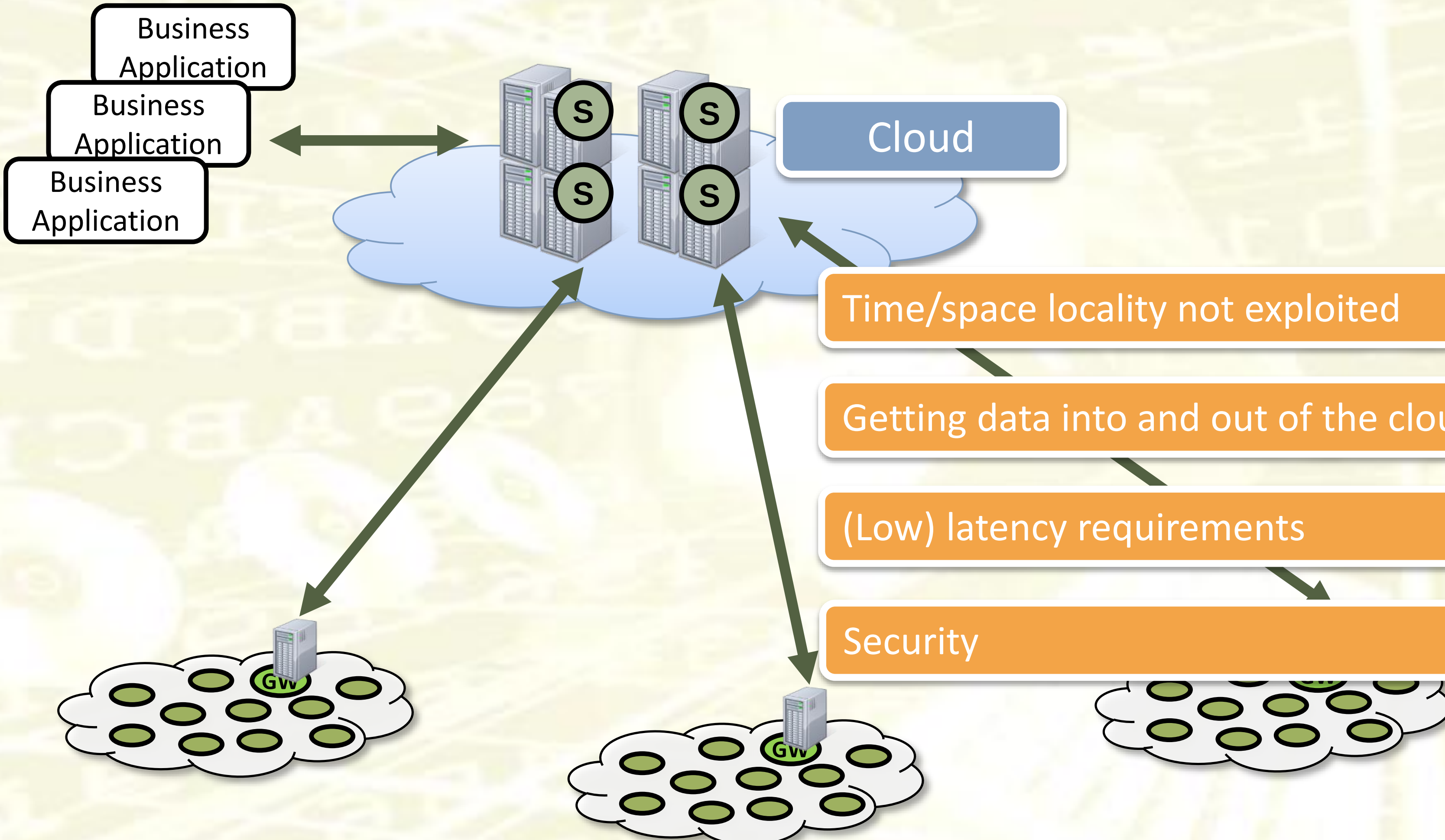
Open Source
OpenIoT



Proprietary
Xively



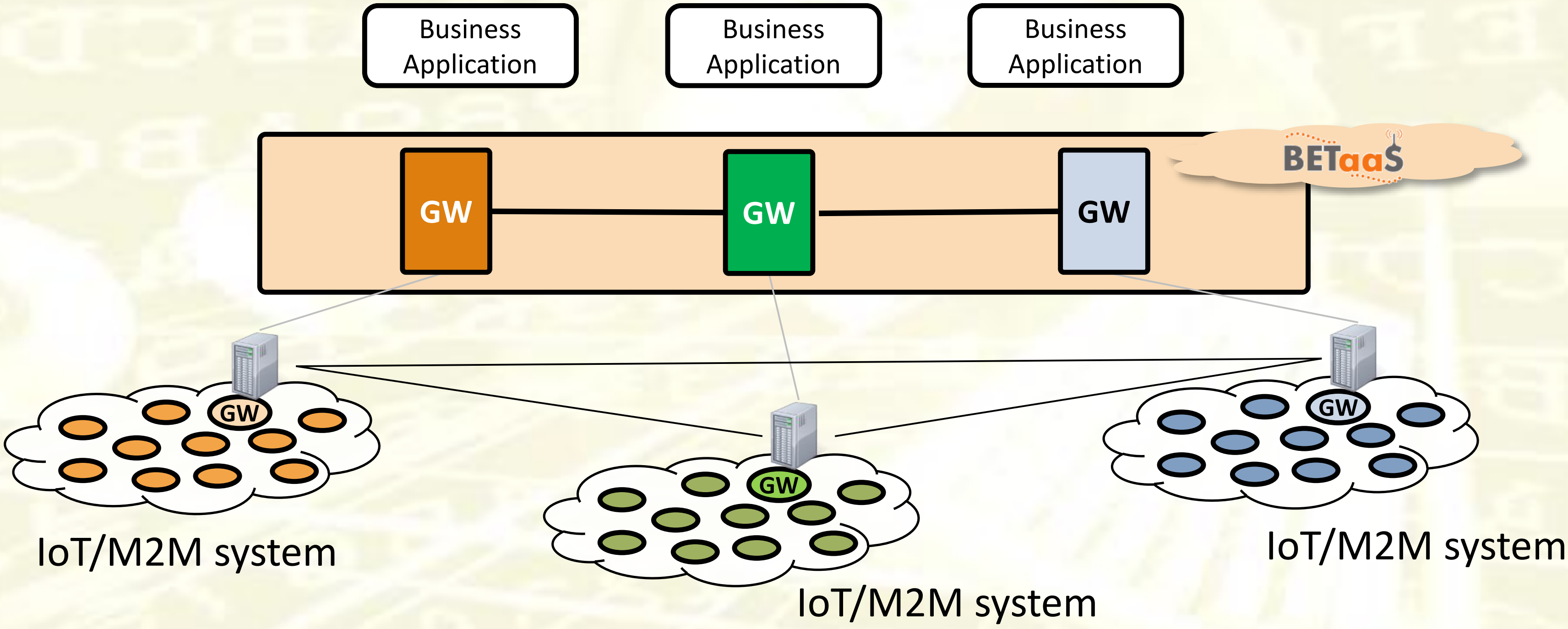
Is the cloud always appropriate?



- Time/space locality not exploited
- Getting data into and out of the cloud
- (Low) latency requirements
- Security

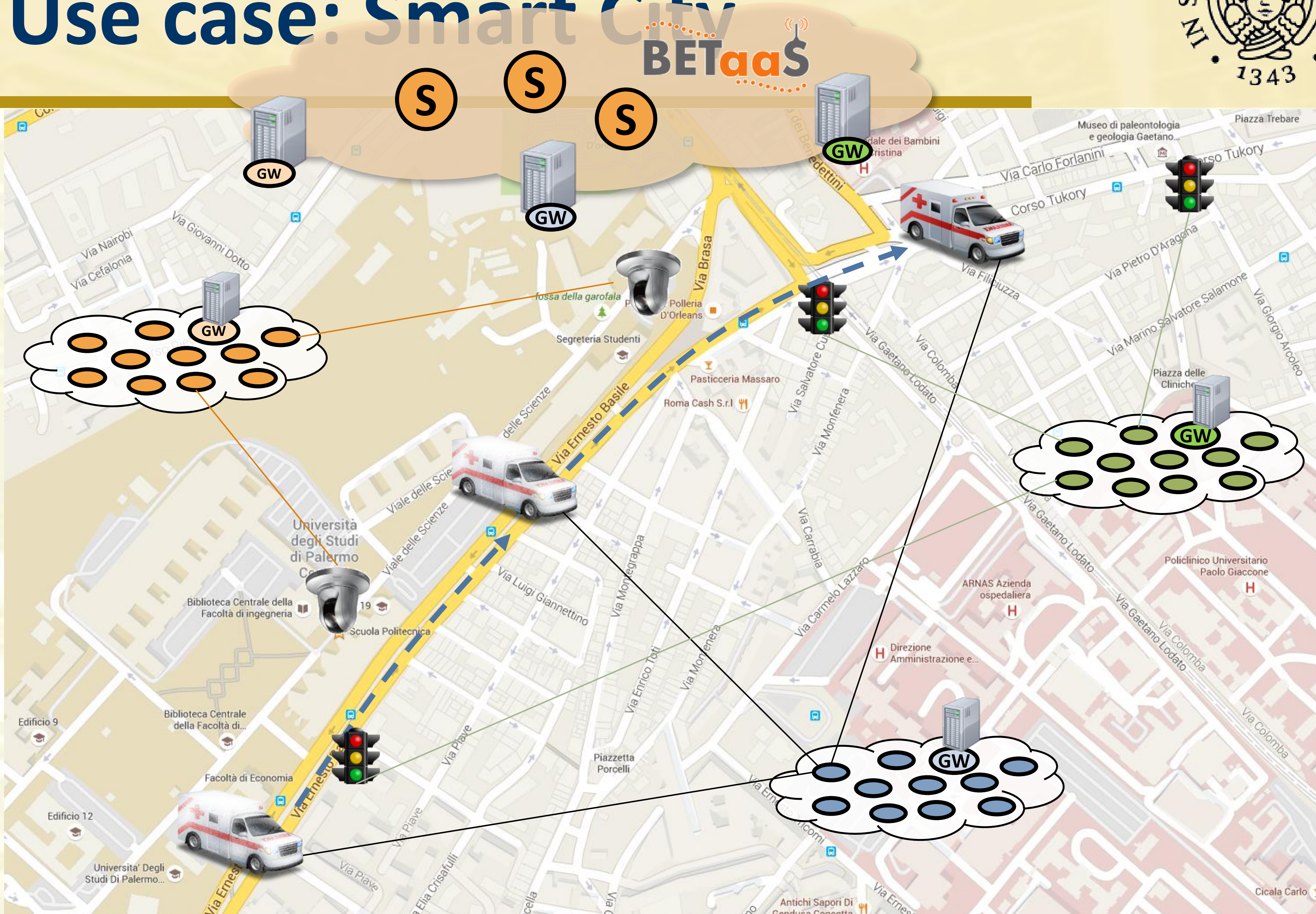
BETaaS approach

- **Move/distribute the intelligence to the edge!!!**
  - *BETaaS gateways*: network devices, set-top boxes, RSUs, …
  - Gateways cooperate to form a runtime (distributed) platform



Use case: Smart Home

Use case: Smart City



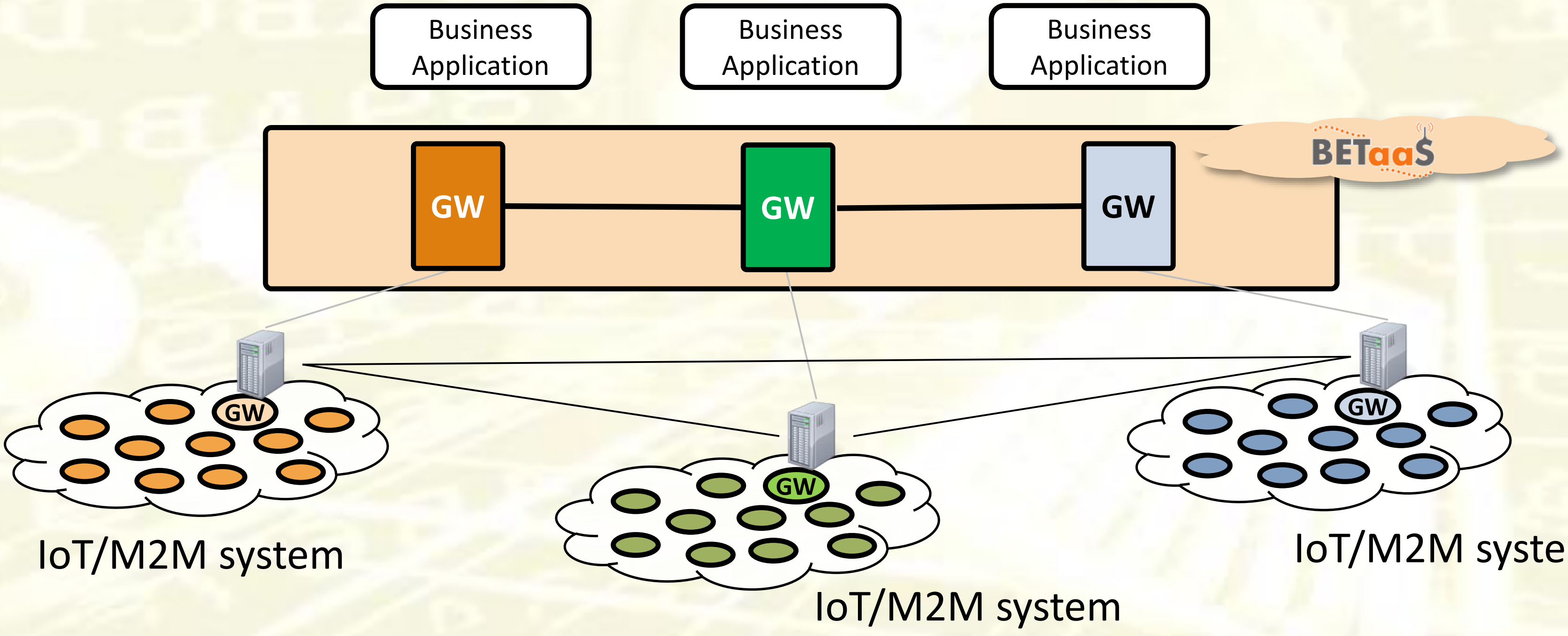
BETaaS approach

- **Move/distribute the intelligence to the edge!!!**

Data storage and processing **close** (in space and time) to where it is generated

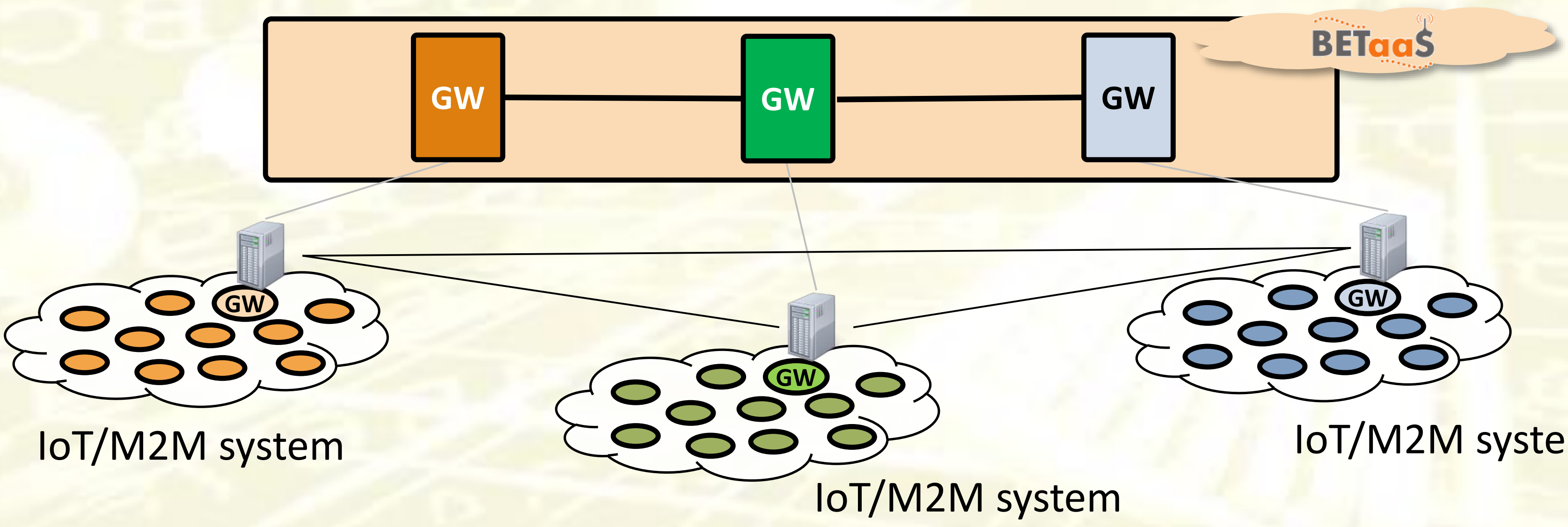
Reduced latency

Resource pooling/optimization



“Local cloud” of gateways

- The set of computational resources hosting the BETaaS runtime environment



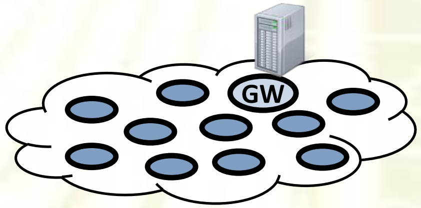
Existing IoT/M2M systems

Heterogeneous physical devices and protocols

Several data formats and structures

No common semantic for resource description

IoT/M2M system

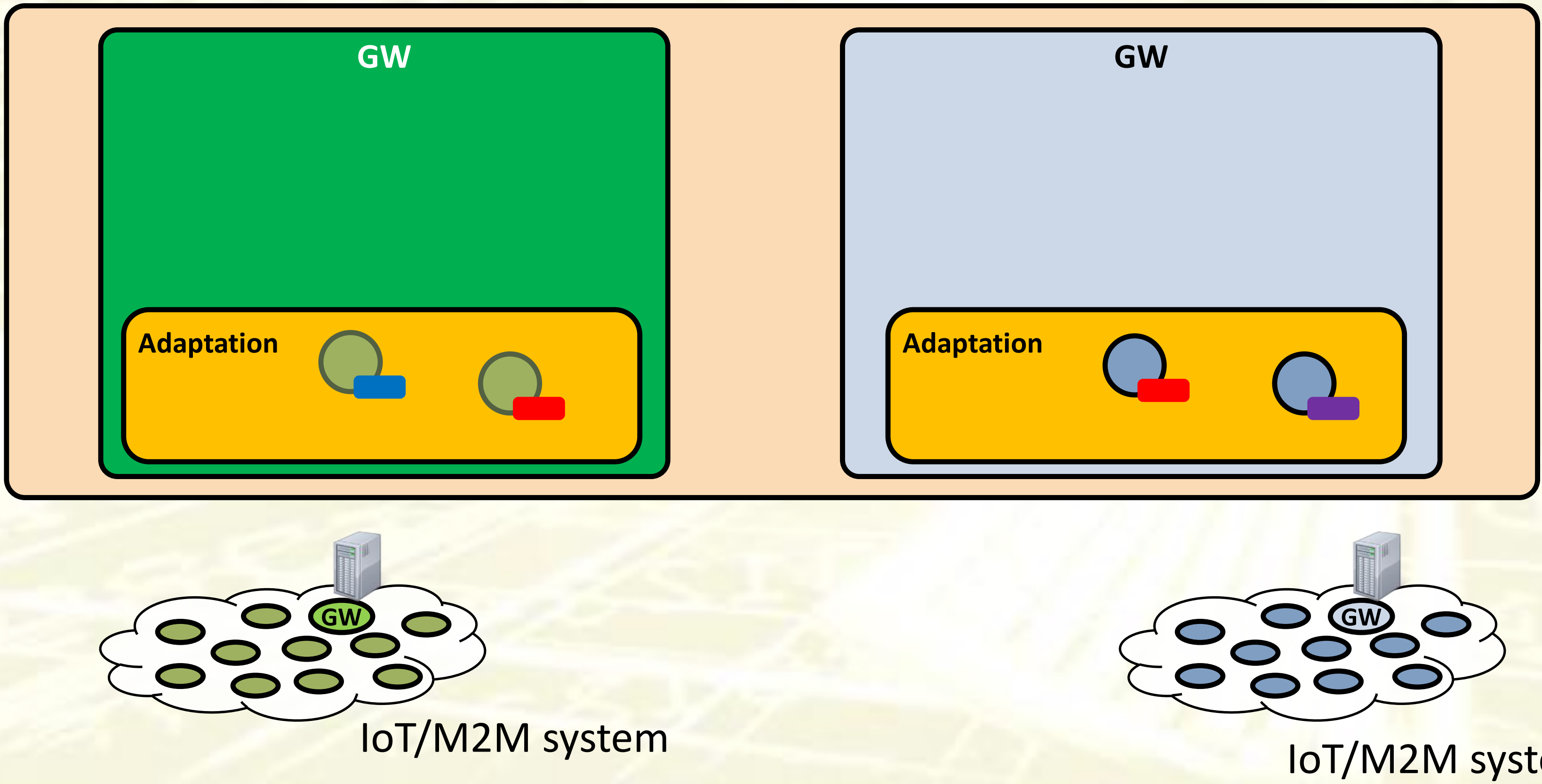


IoT/M2M system

Enable integration

Standardize access through a common interface and **data representation**

Provide a basic set of functionalities by the plugged-in IoT/M2M system

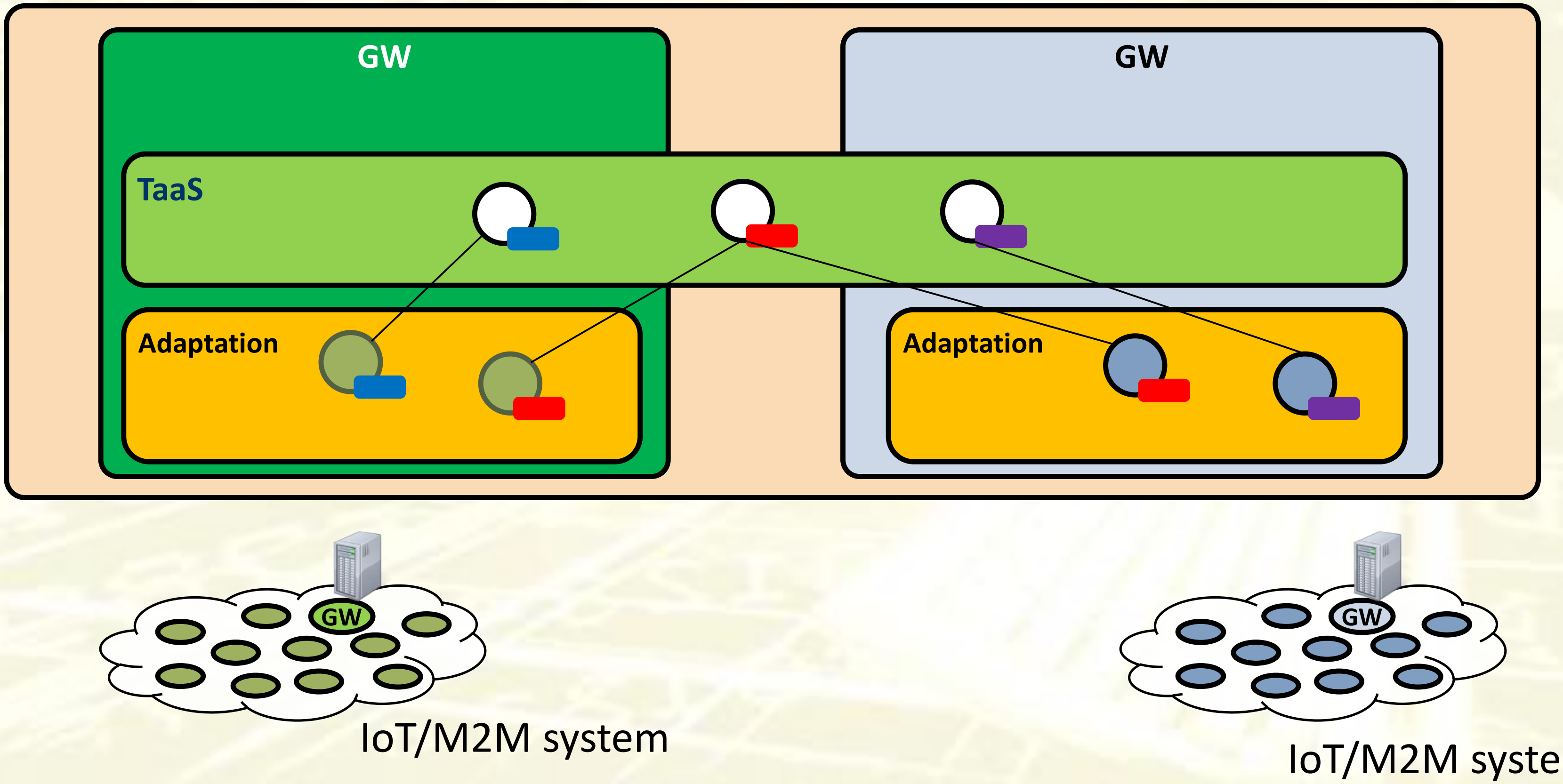


Realize integration: TaaS model

Seamless service-oriented access to things irrespectively of the location

Common semantics to enable context-aware lookup

Support for non-functional requirements (e.g., QoS)

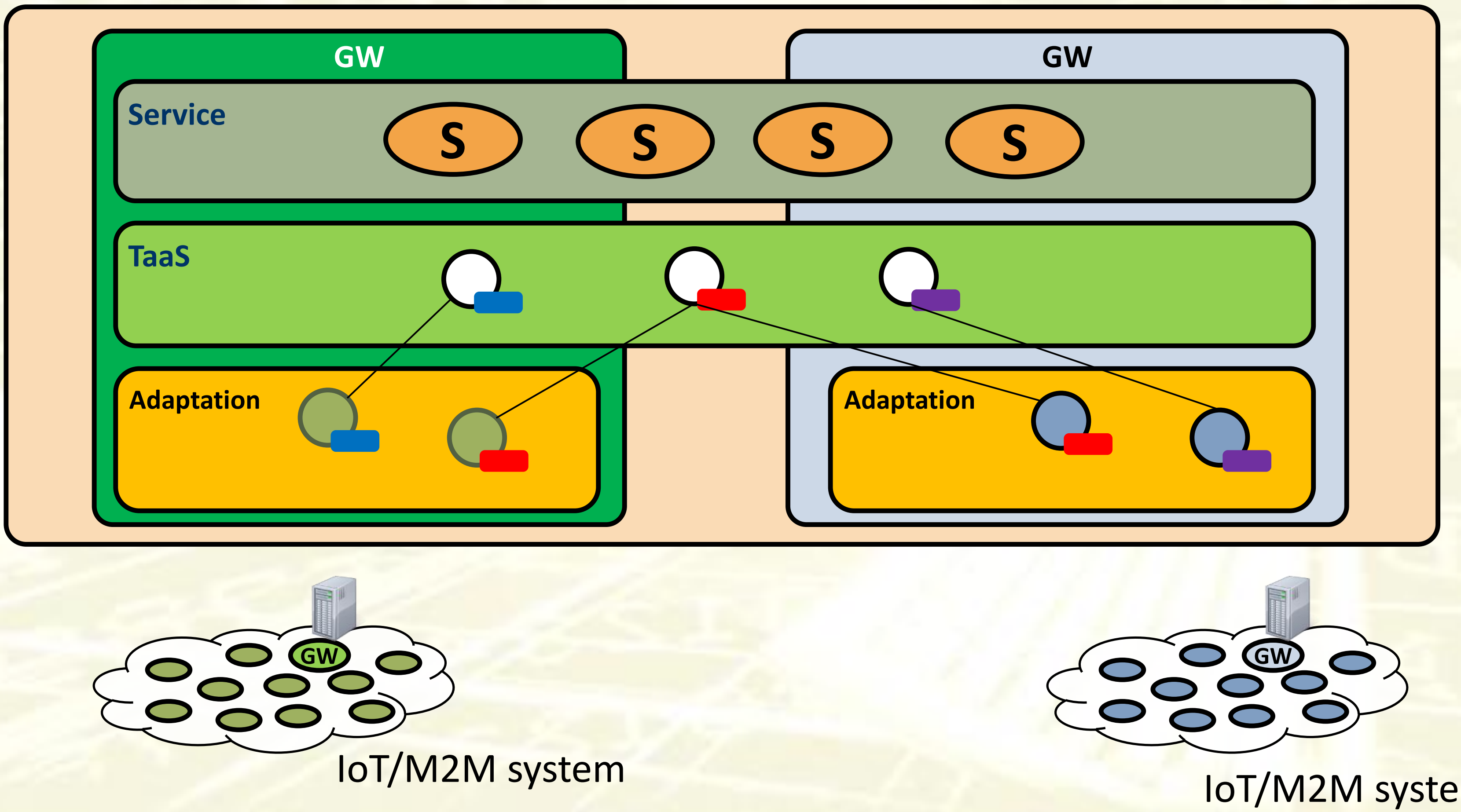


Thing service equivalence

- Different things may provide overlapping information or equivalent functionalities
  - Redundant sensors
    - Industrial automation scenario, e.g., for dependability
    - Large scale deployments (weather info sensors, LoRA, …)
  - Presence information at home: infrared sensors, smart camera, smart thermostat, temperature+humidity, Google, …
  - Road vehicle detection: infrared sensors, smart camera, …
- Equivalence is inferred by semantic reasoning based on contextual information by the Context Manager inside the TaaS layer of BETaaS
  - In the current release, it depends on the service type and location

M2M service deployment

Manage M2M services built on top of TaaS

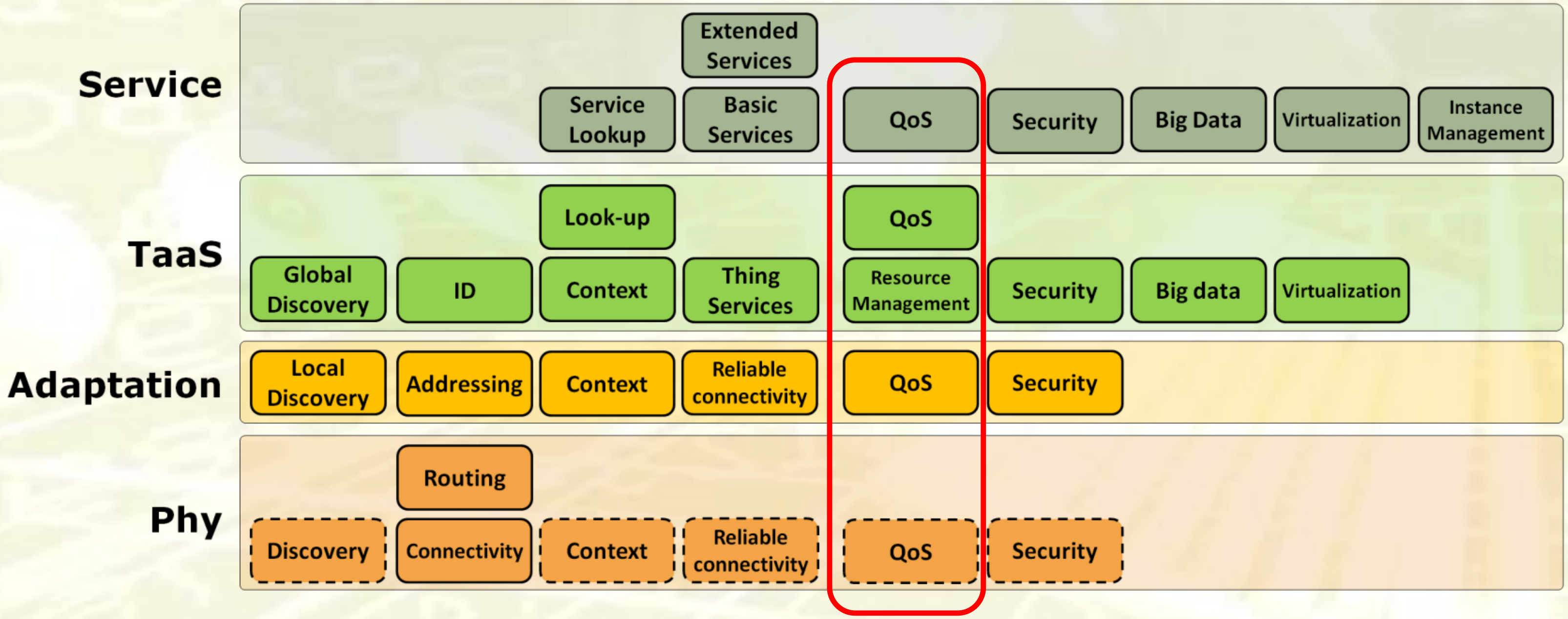


Outline

- IoT horizontal platforms: centralized vs. distributed approach
  - the EU FP7-BETaaS solution: Things-as-a-Service on top of a "local cloud"
- Quality of Service support for M2M applications
  - the BETaaS QoS framework
- Optimized Thing Service selection
  - Exploit multiple TS equivalence to maximize system lifetime: the Agent Bottleneck Generalized Assignment Problem
- Ongoing & future work

BETaaS functional view

- Built-in support for extended features
  - Context-aware lookup
  - QoS, Security, Big Data, Virtualization



Quality of Service support

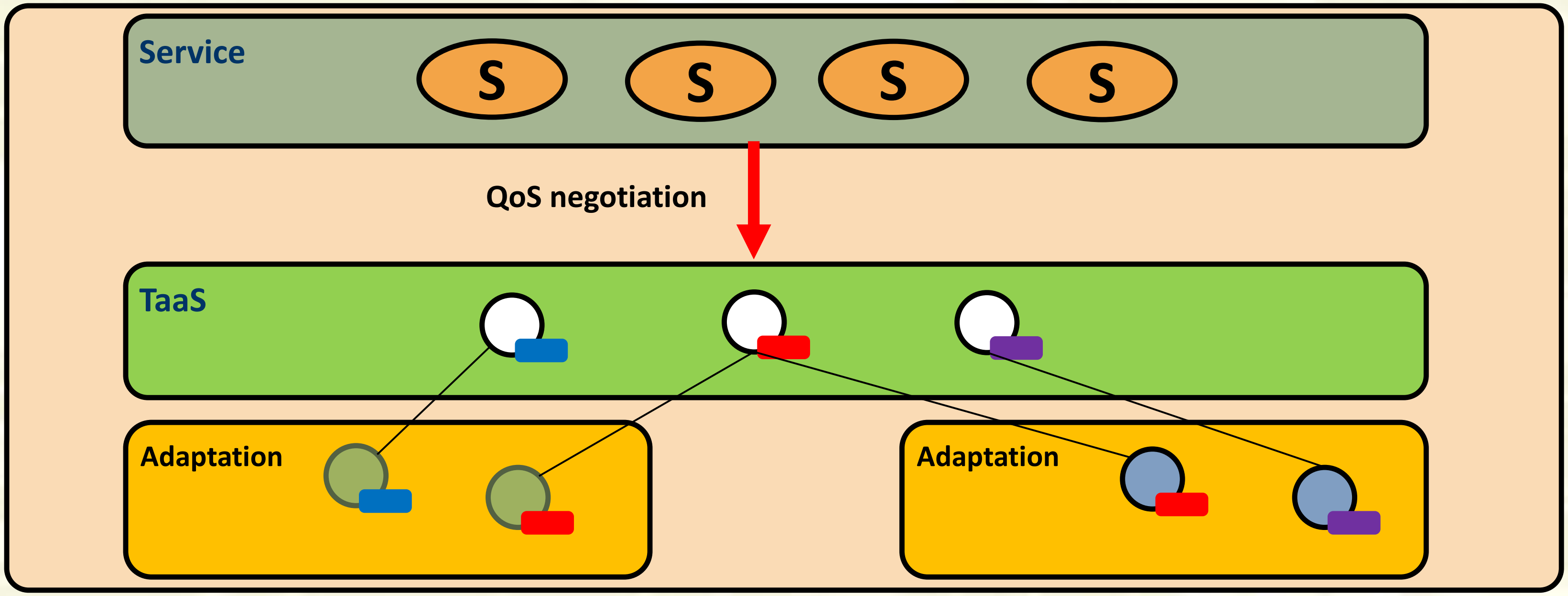
- M2M application scenarios may have very different and unique requirements
- Provide services with associated guarantees on performance
  - Allow applications to negotiate a Service Level Agreement (SLA)
    - QoS model
  - Enforce QoS through the efficient management of resources
    - Resource reservation and optimized allocation
  - Monitor SLA fulfillment

QoS model

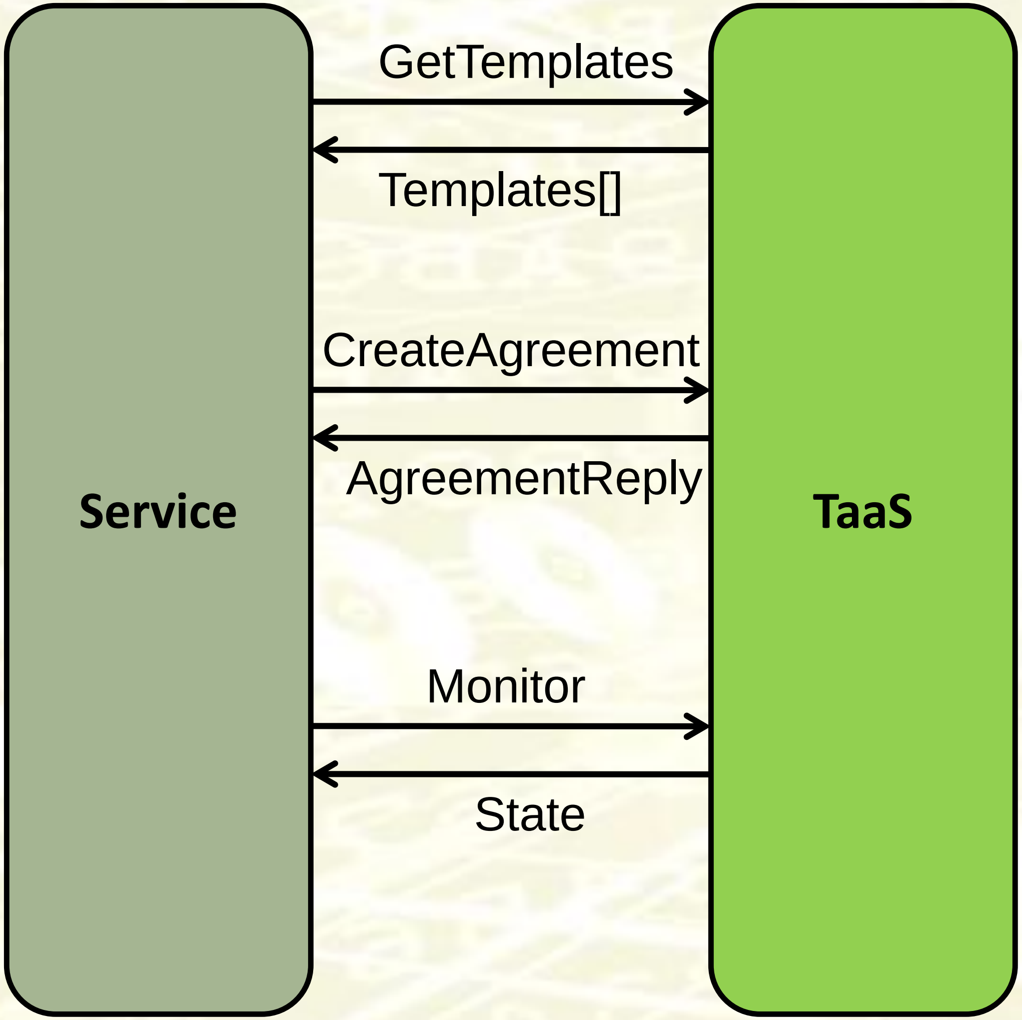
- Classes of service
  - **Real-Time**: Applications with hard response time requirements (deterministic)
    - E.g., Surveillance system, industrial control, healthcare
  - **Assured Services**: Applications with soft response time requirements (e.g., probabilistic)
    - E.g., Road traffic alerts, Vehicle tracking
  - **Best-Effort**: Applications with no time requirements
    - E.g., Meter data collection
- SLA templates defined accordingly

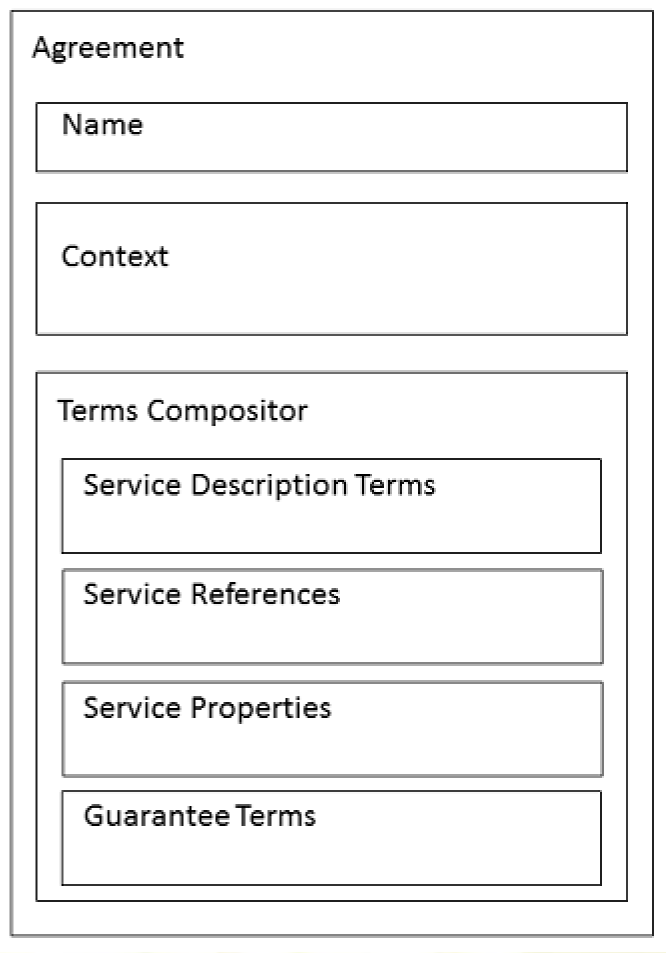
QoS negotiation

- **SLA** negotiation
- **Admission control** based on QoS requirements
- QoS-based **resource reservation**
- **Authorize** Thing Service invocation



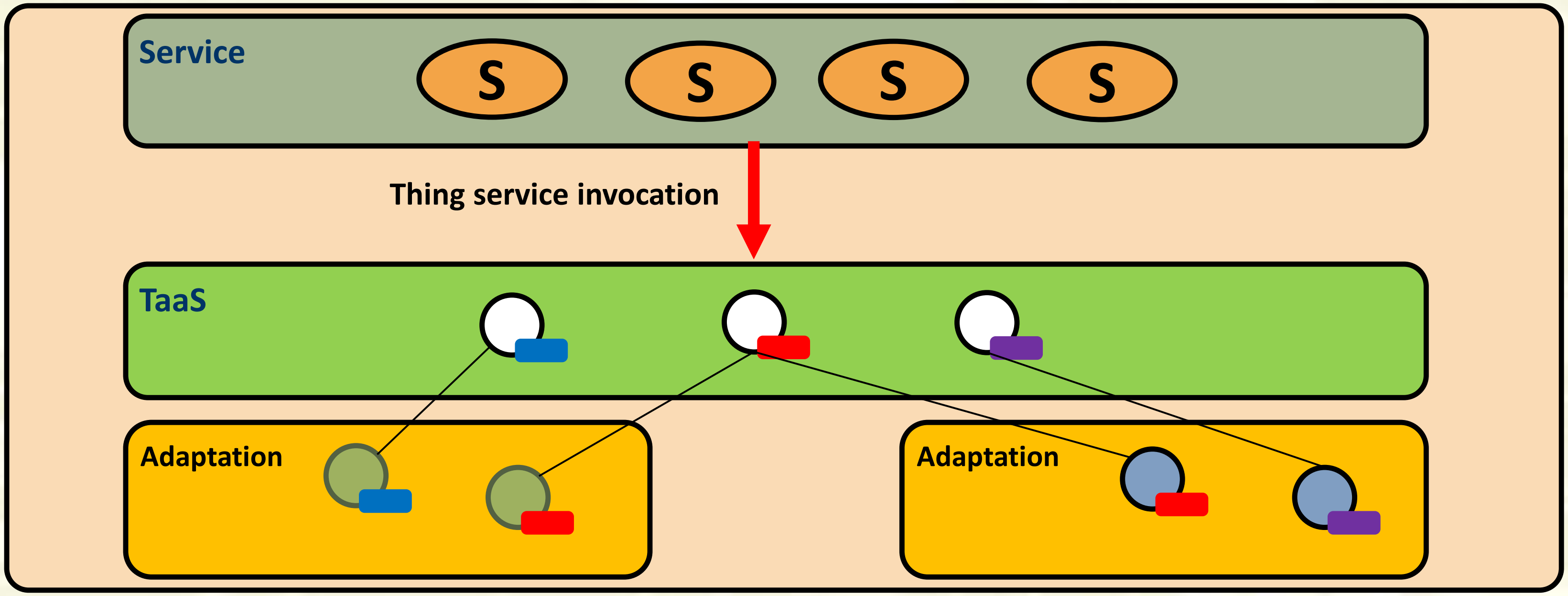
WS-Agreement



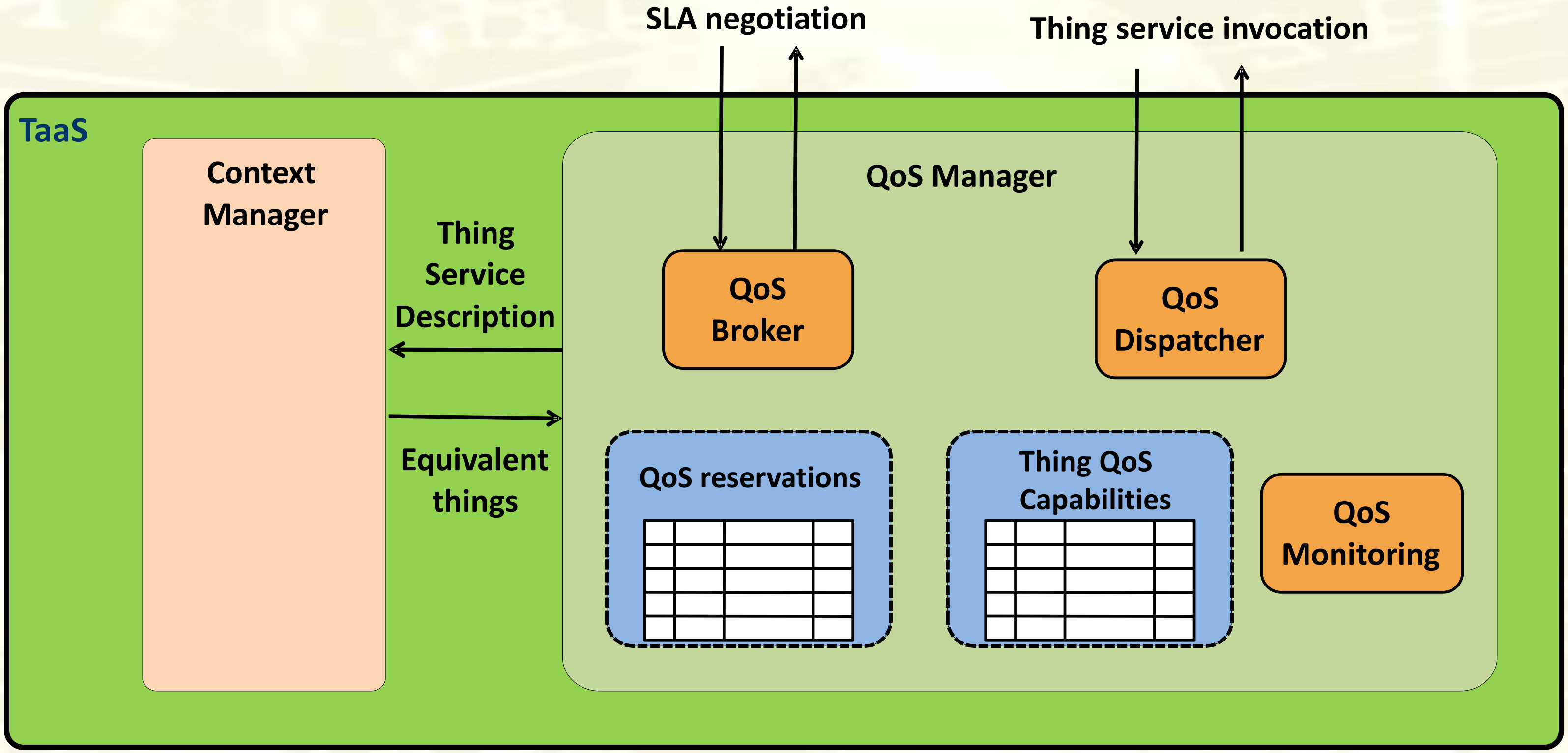


Service invocation

- Select at runtime which thing to allocate the service



QoS manager

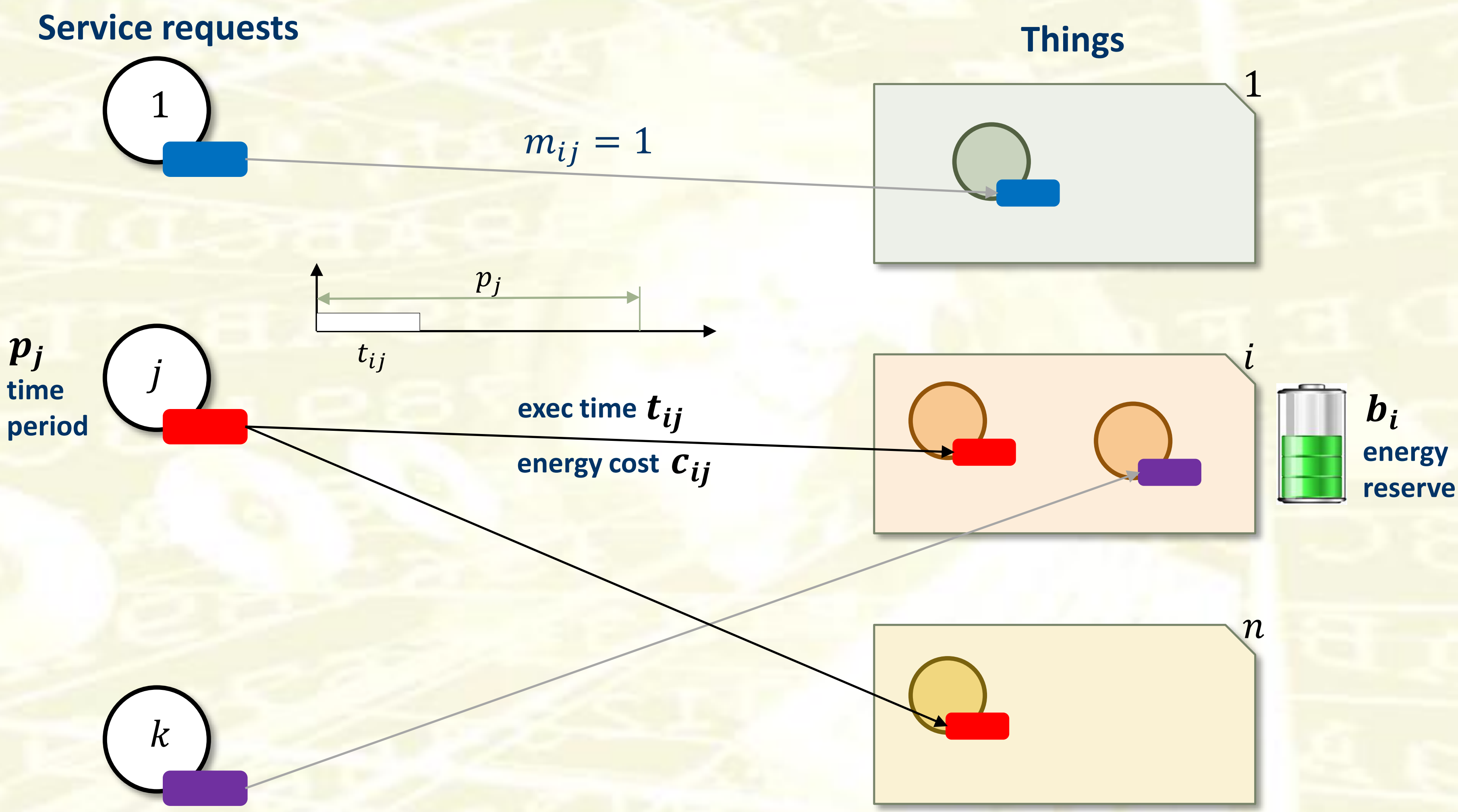


SLA negotiation
Thing service invocation
TaaS
Context Manager
Thing Service Description
Equivalent things
QoS Manager
QoS Broker
QoS Dispatcher
QoS reservations
Thing QoS Capabilities
QoS Monitoring

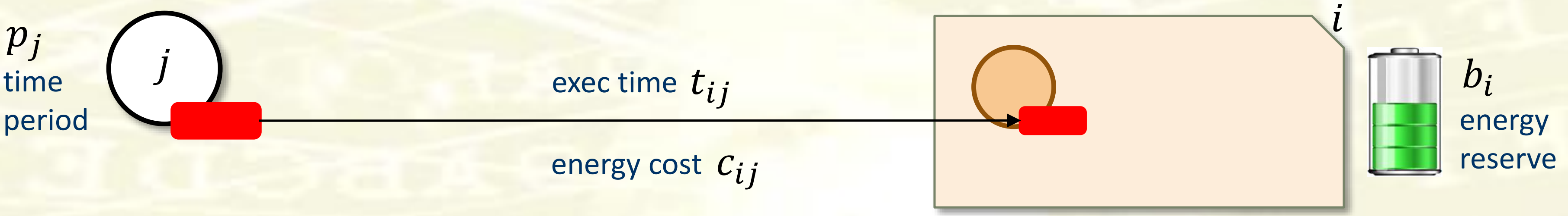
Outline

- IoT horizontal platforms: centralized vs. distributed approach
  - the EU FP7-BETaaS solution: Things-as-a-Service on top of a "local cloud"
- Quality of Service support for M2M applications
  - the BETaaS QoS framework
- Optimized Thing Service selection
  - Exploit multiple TS equivalence to maximize system lifetime: the Agent Bottleneck Generalized Assignment Problem
- Ongoing & future work

Problem formulation



Problem formulation

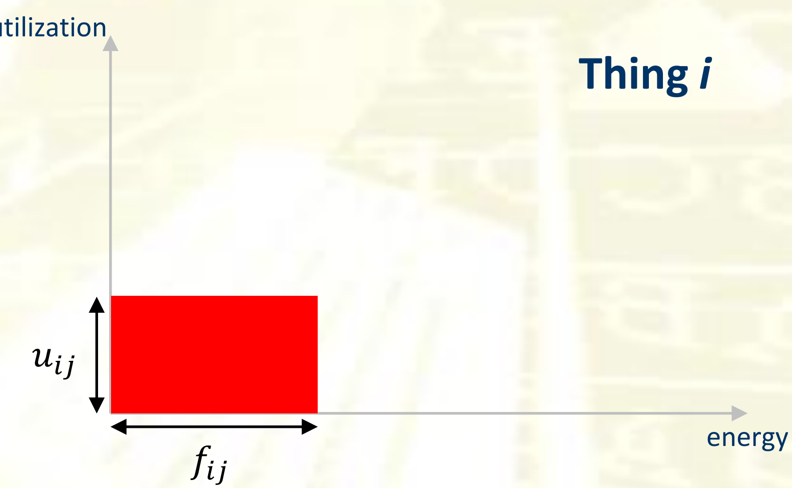


We use a common reference time period *h* to compare energy costs

$$\boldsymbol{h} = lcm(p_j)$$

$$\boldsymbol{u_{ij}} \triangleq \frac{t_{ij}}{p_j}$$ utilization

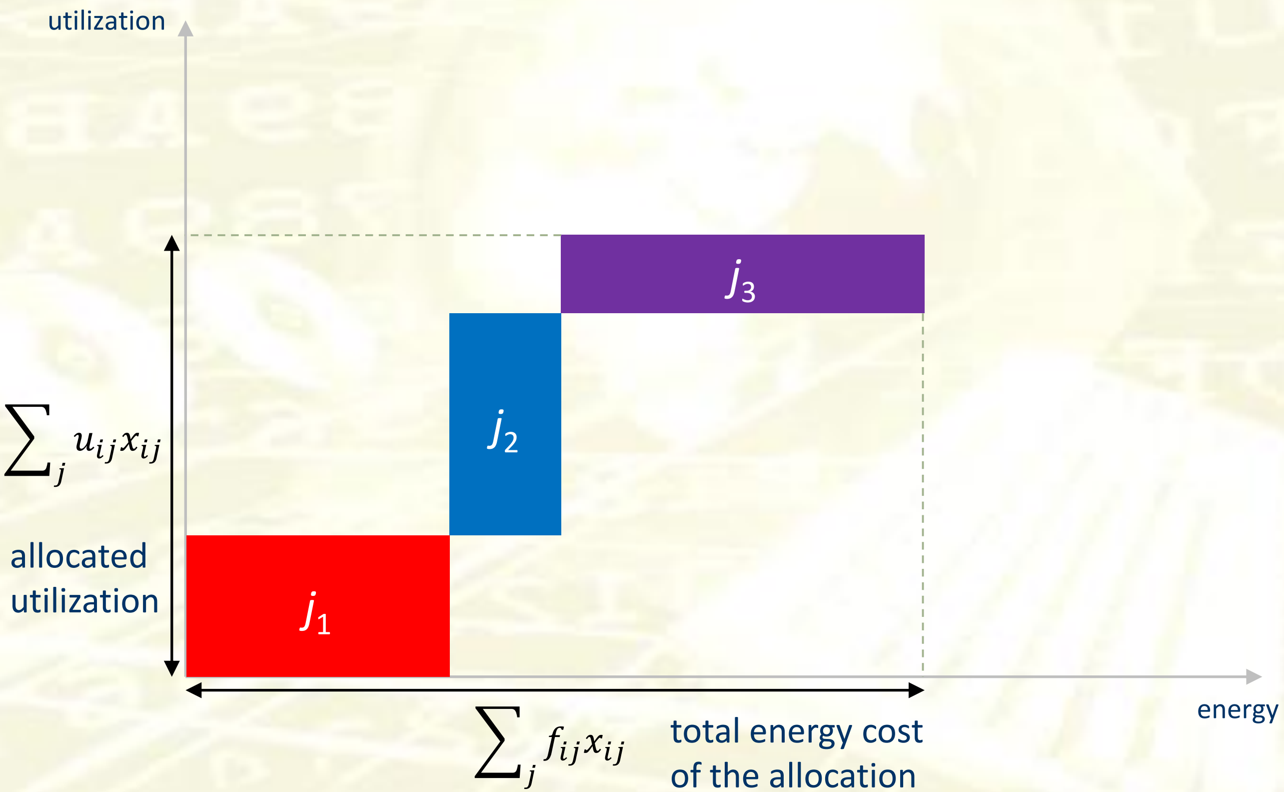
$$\boldsymbol{f_{ij}} \triangleq \left(\frac{h}{p_j} \cdot c_{ij}\right) / b_i$$ fraction of available energy needed by service *j* on thing *i* to complete all executions over the common reference period *h*



Problem formulation

Problem variables

$\boldsymbol{x_{ij}} = 1$ if request j is allocated to thing i, 0 otherwise



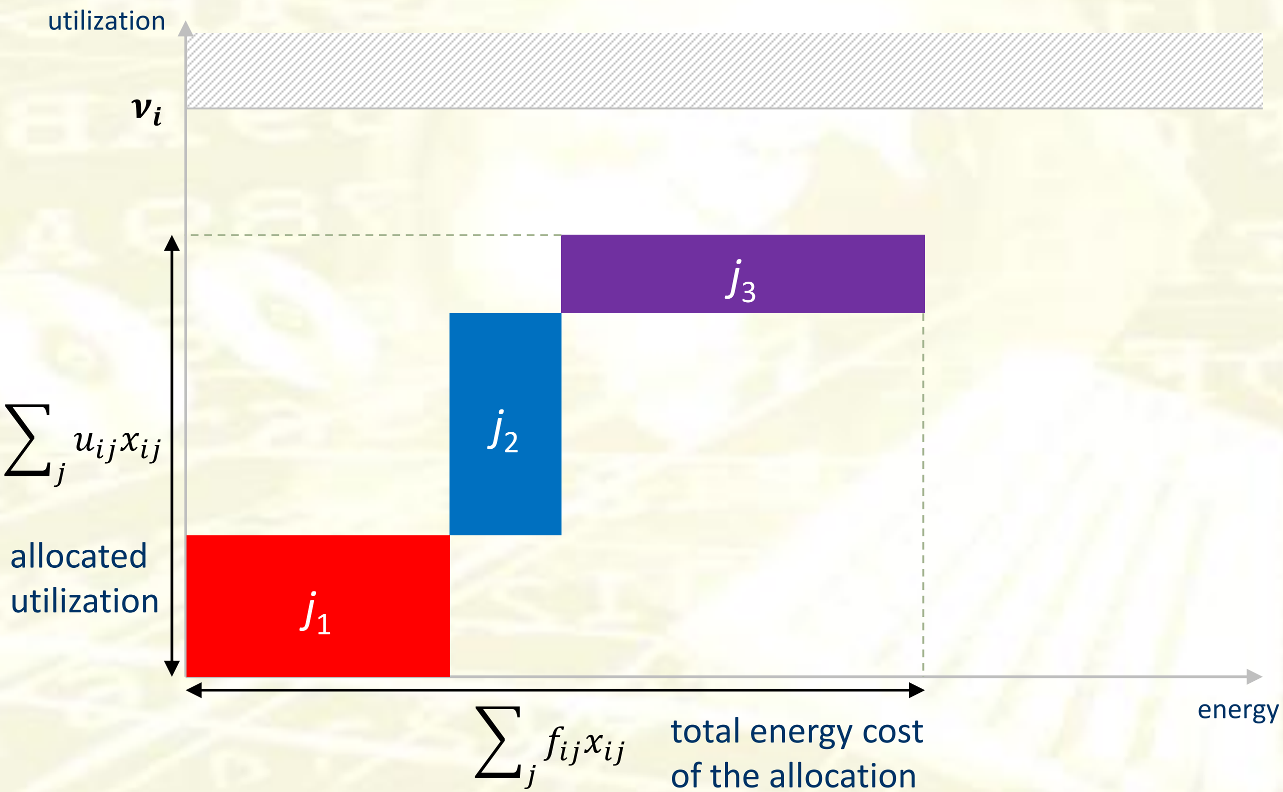
Problem formulation

Constraints? **Schedulability**

$$\sum_j u_{ij} x_{ij} < \boldsymbol{\nu_i}$$

$$\boldsymbol{\nu_i} = s_i \left(2^{\frac{1}{s_i}} - 1\right)$$

$$s_i = \sum_j x_{ij}$$

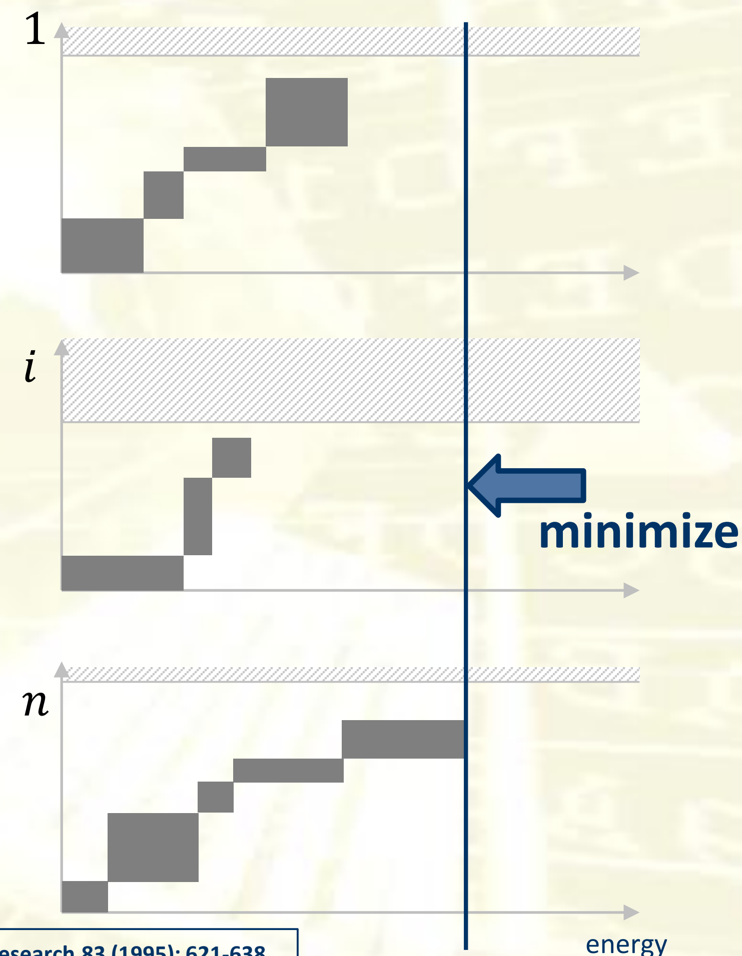


Thing *i*

Problem formulation

Objective? **Minimize the maximum total energy cost *per* thing**

$$\min\left(\max_i \sum_j f_{ij}x_{ij}\right)$$



Bottleneck Generalized Assignment Pr.

Minimax objective

Multiple service requests assigned to one thing

$\max_i f_{ij}x_{ij}$ **Task BGAP [Martello,Toth 1995]**

$\max_i \sum_j f_{ij}x_{ij}$ **Agent BGAP**

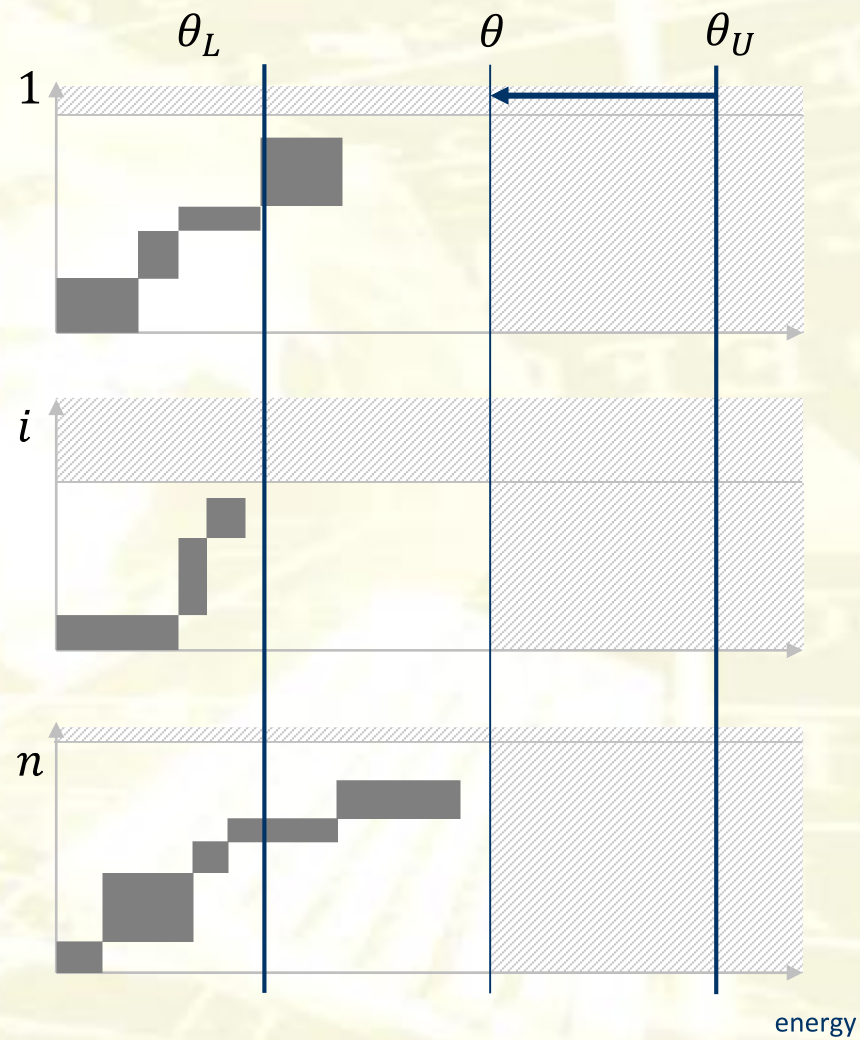
S. Martello and P. Toth "The bottleneck generalized assignment problem." European Journal of Operational Research 83 (1995): 621-638

Solution

- Solving for minimax

  1. heuristically search for a **feasible solution** of a value lowest than a **threshold ϑ**

  2. heuristically search for the **lowest value ϑ** for which a feasible solution is found

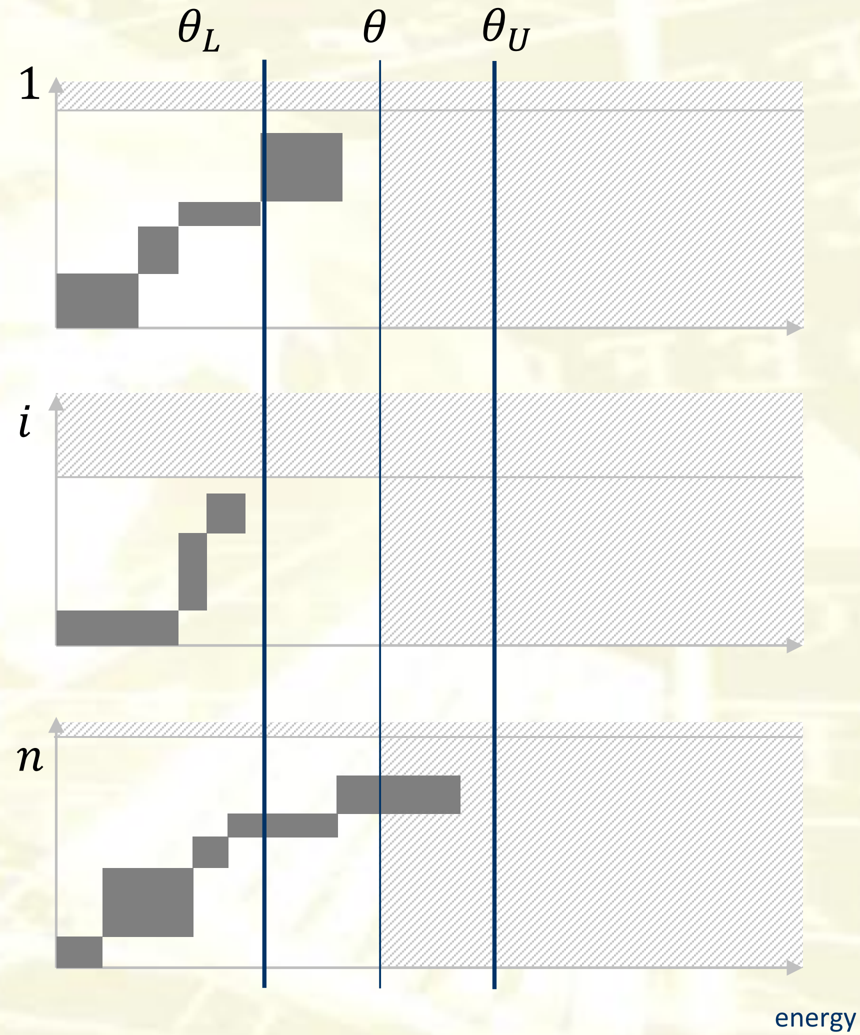


Solution

- Solving for minimax

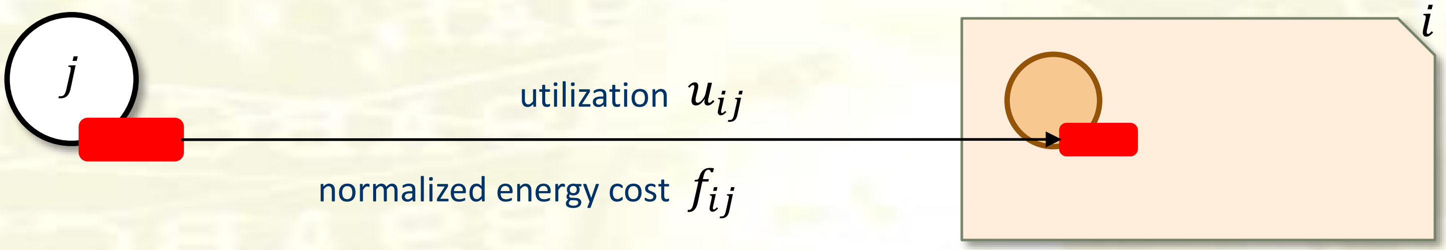
  1. heuristically search for a **feasible solution** of a value lowest than a **threshold *ϑ***

  2. heuristically search for the **lowest value *ϑ*** for which a feasible solution **is found**



Solution

Find a **good** feasible solution (inspired by TBGAP)



Set a ***fixed*** priority $\boldsymbol{p_{ij}}$ measuring the ***desirability*** of allocating service *j* to thing *i*

priority p_{ij}

Many choices available

$p_{ij} = u_{ij}$ Largest job first

$p_{ij} = f_{ij}$ Highest (energy) cost first

$p_{ij} = -f_{ij}$ Lowest (energy) cost first

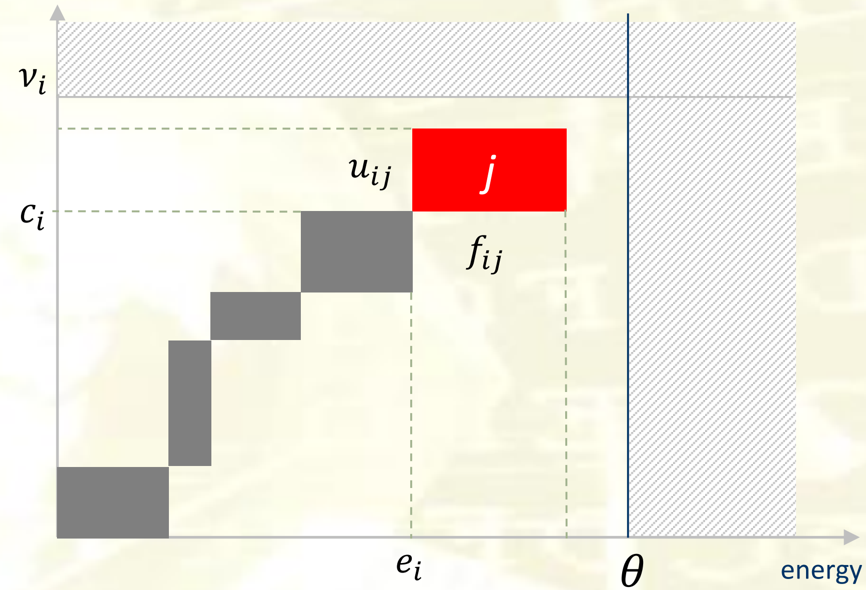
Solution

Find the next request to allocate ***j****

Calculate, for each *j*

$$\boldsymbol{F_j} = \{\boldsymbol{p_{ij}}: m_{ij} = 1; c_i + u_{ij} < v_i; e_i + f_{ij} < \theta\}$$

$$\boldsymbol{\delta_j} = \max F_{ij} - \max_2 F_{ij}$$

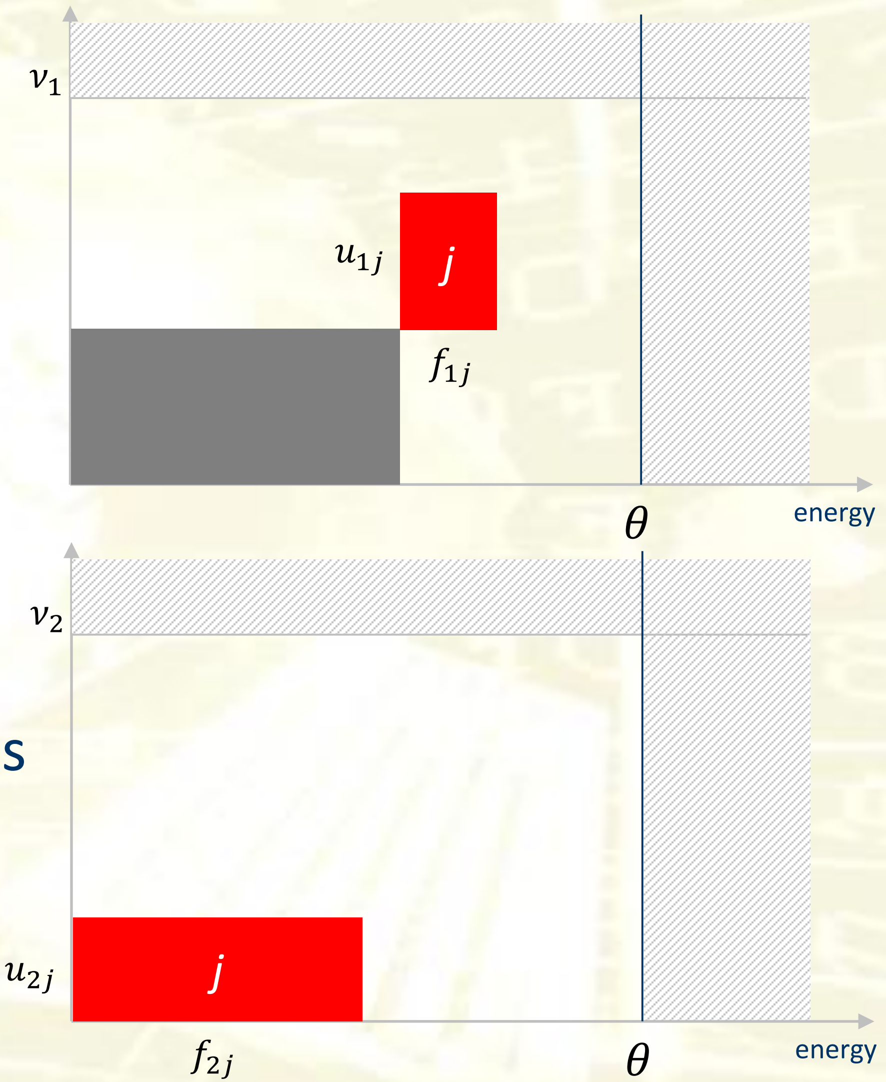


Allocate $\boldsymbol{j}^* = \text{argmax}\, \delta_j$ to $\boldsymbol{i}^* = \text{argmax}\, F_{j^*}$

Issue with fixed priorities

Example

$$p_{ij} = -f_{ij}$$



Fixed priorities (*desirability*) may lead to **increase the energy consumption** on the bottleneck

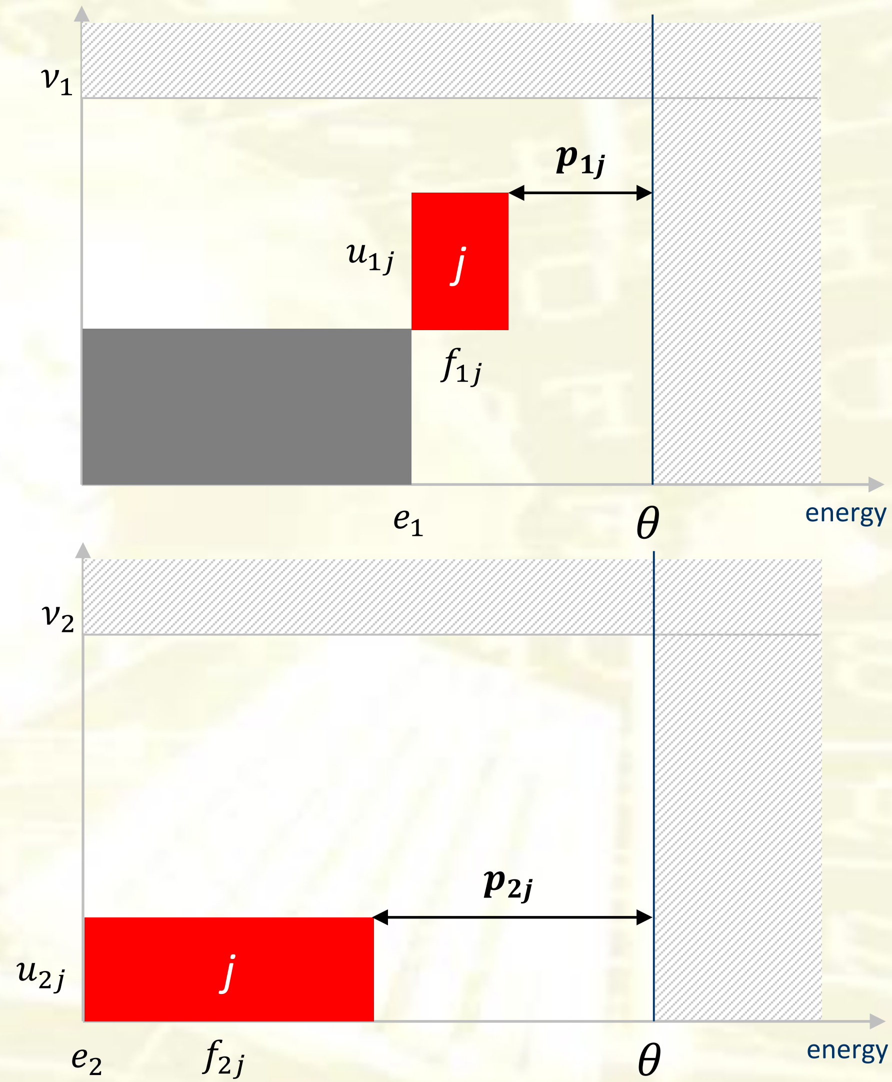
The reason is that desirability depends on the current working solution, i.e., should be **dynamic**

Dynamic priorities

Set priorities based on residual energy on each thing according to the current solution

$$p_{ij} = \theta - (e_i + f_{ij})$$

RTTA: Real-Time Thing allocation algorithm



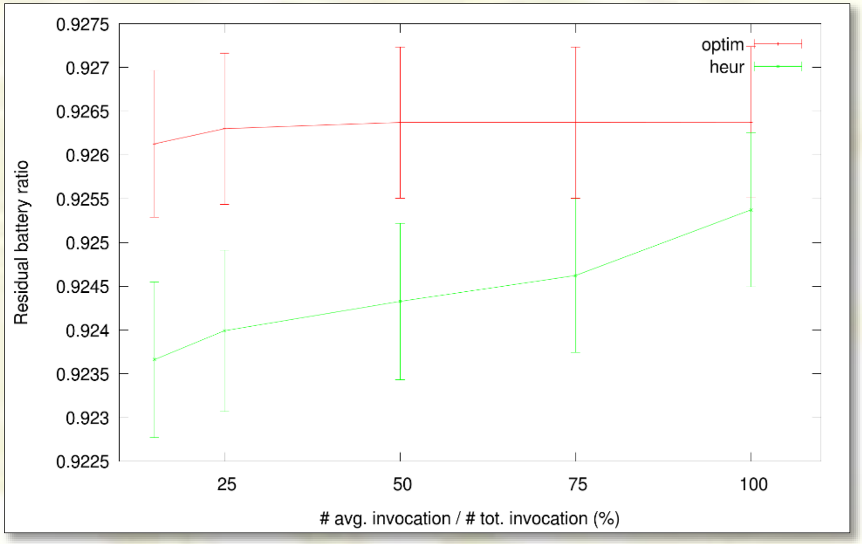
Numerical evaluation

- C++ implementation of RTTA vs. standard optimization solver (IBM iLOG CPLEX)
- Randomly generated input

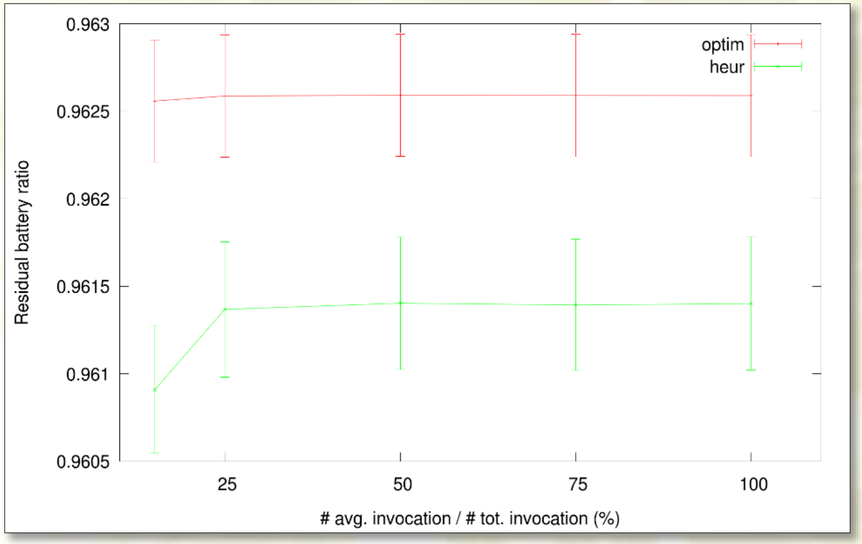
Parameter	Range
Period	10 – 100 s - step 10 s
Initial battery level	50 – 25 mJ - step 5 mJ
Execution cost	$2 \cdot 10^{-4} – 6 \cdot 10^{-4}$ mW
Execution time	7 – 22.5 ms

Numerical results

50 things, 500 requests



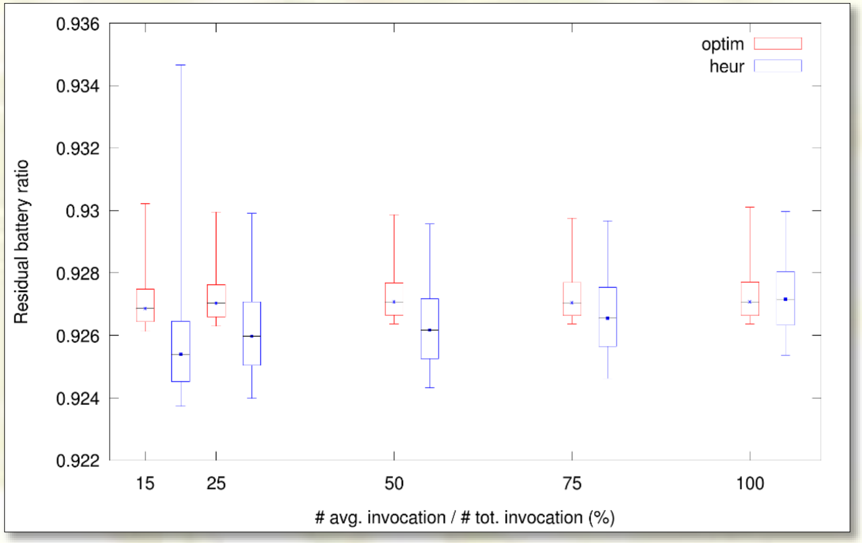
100 things, 500 requests



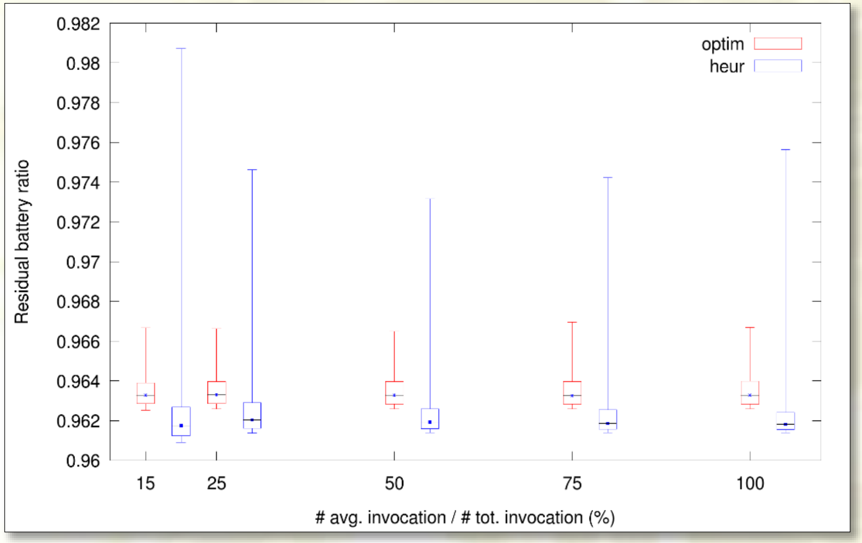
- Residual battery (average) after one period
  - RTTA aims at maximizing the minimum

Numerical results

50 things, 500 requests



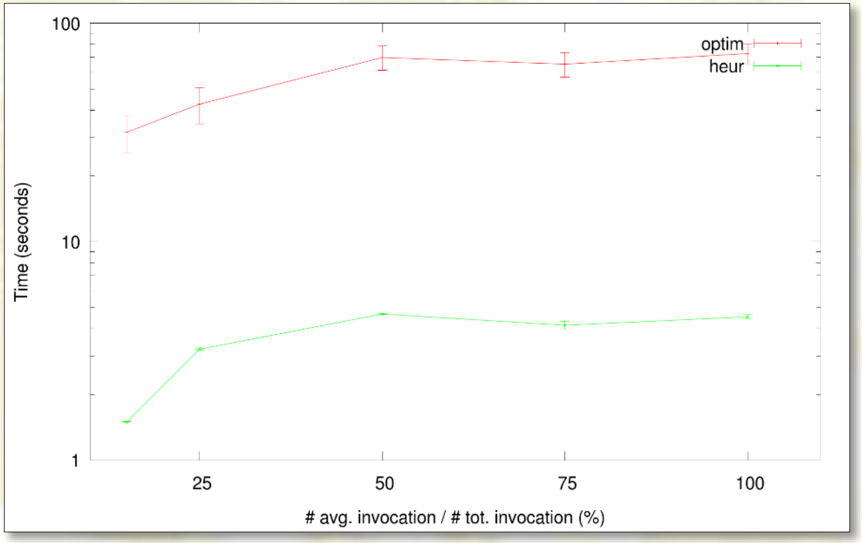
100 things, 500 requests



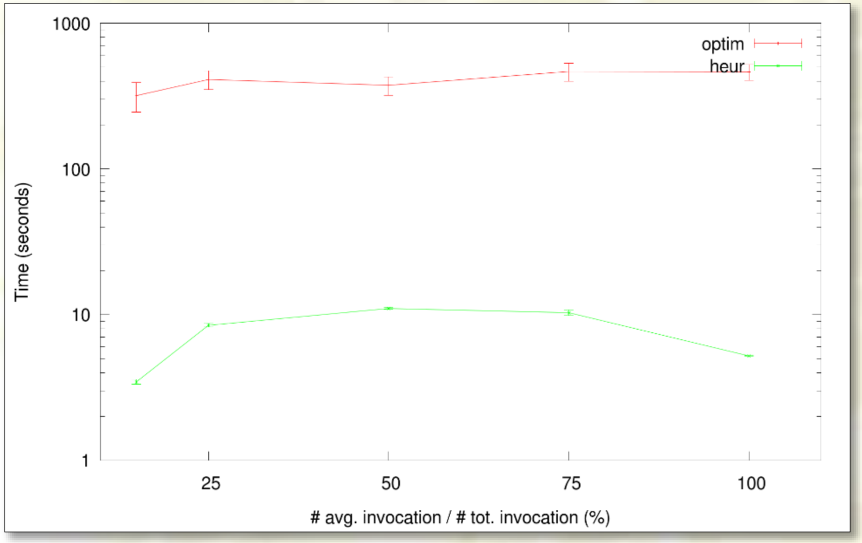
- Residual battery (distribution) after one period

Numerical results

50 things, 500 requests



100 things, 500 requests

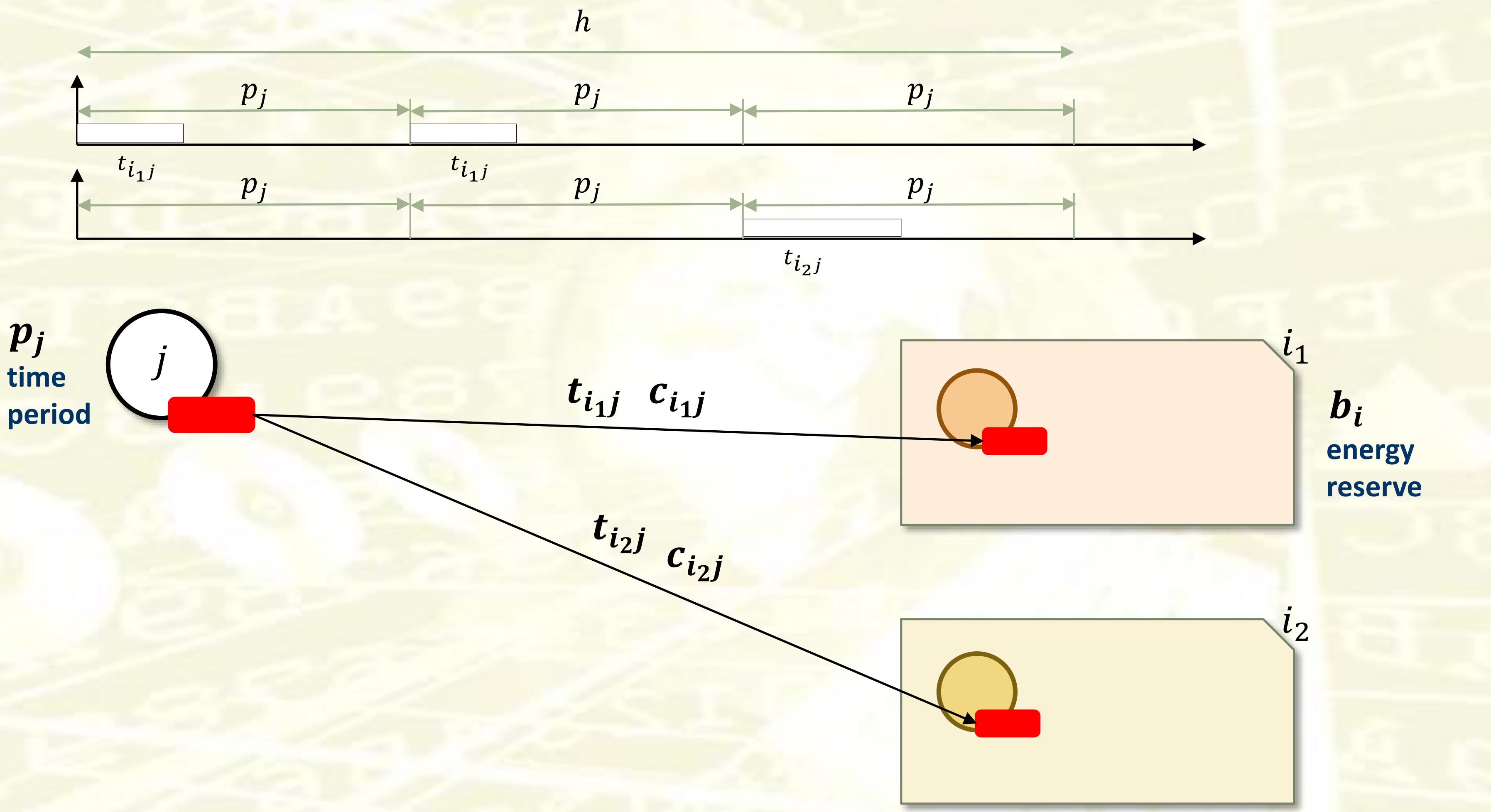


- Computation time

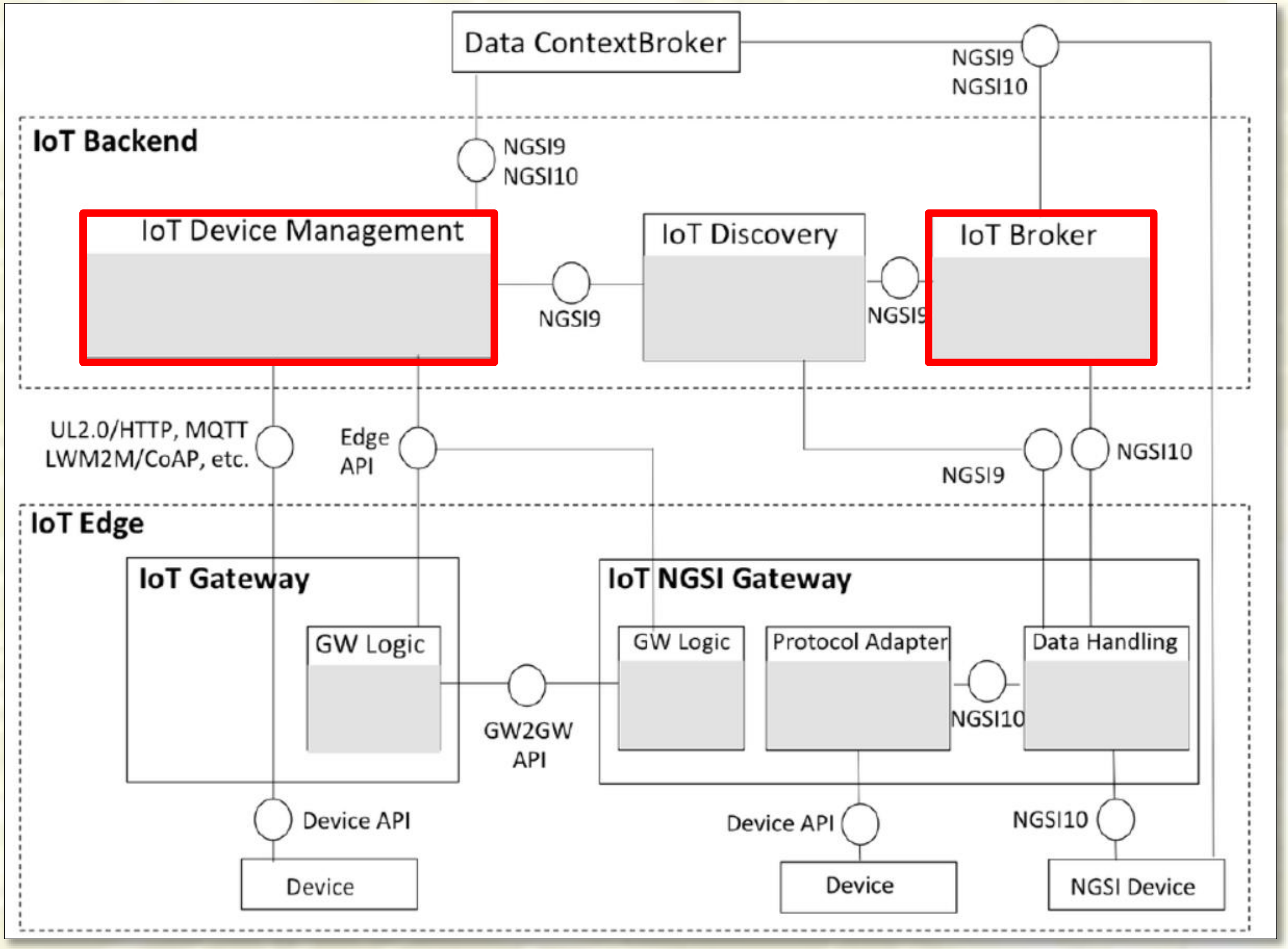
Outline

- IoT horizontal platforms: centralized vs. distributed approach
  - the EU FP7-BETaaS solution: Things-as-a-Service on top of a "local cloud"
- Quality of Service support for M2M applications
  - the BETaaS QoS framework
- Optimized Thing Service selection
  - Exploit multiple TS equivalence to maximize system lifetime: the Agent Bottleneck Generalized Assignment Problem
- Ongoing & future work

Split allocation



QoS support integration in FIWARE



References

- VV.AA., **BETaaS platform – a Things as a Service Environment for future M2M marketplaces**, *EAI Endorsed Transactions on Cloud Systems*, 1:1, pp. 1-9, Feb. 2015
- E. Mingozzi, G. Tanganelli, C. Vallati, **A framework for Quality of Service support in Things-as-a-Service oriented architectures**, *J. of Communication, Navigation, Sensing and Services (CONASENSE)*, 1:2, pp. 105-128, Aug. 2014
- G. Tanganelli, C. Vallati, E. Mingozzi, **Energy-Efficient QoS-aware Service Allocation for the Cloud of Things,** *IEEE CloudCom 2014*, Singapore, Dec. 15-18, 2014
- E. Mingozzi, G. Tanganelli, C. Vallati, B. Martínez, I. Mendia, M. González-Rodríguez, **Exploiting Semantic-based Thing Equivalence in a QoS-aware M2M Fog Platform**, *IEEE WF-IoT 2015*, Milan, IT, Dec. 14-16, 2015

Thanks!

Enzo Mingozzi
Associate Professor @ University of Pisa
e.mingozzi@iet.unipi.it